PRIOR Document No.: 98-0040-K

DIINS ADM Development

SOFTWARE DESIGN DOCUMENT

Version 1.0
31 July 1998

Prepared by: _____
Gary Jackson
Senior Systems Analyst

Reviewed by: _____
Tom Davis
Project Manager

Approved by: _____
Gerry Dawson
Quality Assurance Engineer

102-02437/

PRIOR Document No.: 98-0040-K

DIINS ADM Development

SOFTWARE DESIGN DOCUMENT (SDD)

Version 1.0
31 July 1998

Prepared for:

Prepared for:
Defence Research Establishment Ottawa
Department Of National Defence
Ottawa, Ontario K1A 0Z4

Prepared by:

PRIOR Data Sciences Ltd.
240 Michael Cowpland Drive
Kanata, Ontario
K2M 1P6

REVISION NOTES

VERSION	DATE	DESCRIPTION
0.1	13 March 98	First Draft
1.0	31 July 1998	First Release

TABLE OF CONTENTS

Section		Page
1.	SCOPE	1-1
1.1	Identification	1-1
1.2	System Overview	1-1
1.3	Document Overview	1-2
1.4	Acknowledgments	1-2
2.	REFERENCED DOCUMENTS	2-1
3.	CSCI-WIDE DESIGN DECISIONS	3-1
4.	CSCI ARCHITECTURAL DESIGN	4-1
4.1	CSCI Components	4-1
4.1.1	High Level Architecture	4-1
4.1.1.1	Device Drivers	4-3
4.1.1.2	Real Sensor	4-3
4.1.1.3	Bounded Buffer	4-4
4.1.1.4	Prefilter/Filter	4-4
4.1.1.5	FDL	4-4
4.1.1.6	Data Collector	4-5
4.1.1.7	NMEA Encoder	4-5
4.1.1.8	Task Master	4-5
4.1.2	Human Computer Interface, Control and Display CSC	4-5
4.1.3	Device Drivers CSC	4-8
4.1.4	Encoders CSC	4-9
4.1.5	Data Collector CSC	4-10
4.1.6	Sensor CSC	4-11
4.1.7	Bounded Buffer CSC	4-11
4.1.8	Prefilter/Filter CSC	4-12
4.1.9	Failure Detection and Isolation Controller CSC	4-12
4.1.10	Task Master CSC	4-12
4.1.11	System Configuration Controller CSC	4-13
4.2	Concept of Execution	4-13
4.2.1	DIINS Tasking Architecture	4-13
4.2.1.1	Task Master	4-15
4.2.1.2	Real Sensors	4-15
4.2.1.3	Simulated Sensors	4-16
4.2.1.4	Prefilter/Filter	4-17
4.2.1.4.1	Low Rate Data Prefilter/Filter	4-17
4.2.1.4.2	High Rate Attitude Data Filter	4-19
4.2.1.5	Failure Detection and Isolation	4-20
4.2.1.6	Data Logging	4-21

Section		Page
4.2.1.7	Data Display	4-21
4.2.1.8	Performance Monitor	4-22
4.2.1.9	DIINS System Behavior	4-22
4.3	Interface Design	4-38
4.3.1	Interface Identification and Diagrams	4-38
5.	CSCI DETAILED DESIGN	5-1
5.1	Overall Software Dependencies	5-1
5.2	Human Computer Interface, Control and Display CSC	5-3
5.3	Task Master CSC	5-4
5.4	Device Drivers CSC	5-4
5.5	Encoders CSC	5-4
5.6	Data Collector CSC	5-4
5.7	Sensor CSC	5-9
5.8	Bounded Buffer CSC	5-14
5.9	Prefilter/Filter CSC	5-14
5.9.1	Low Rate Data Prefilter/Filter	5-14
5.9.2	High Rate Attitude Data Filter	5-17
5.10	Failure Detection and Isolation Controller CSC	5-18
5.11	System Configuration Controller CSC	5-18
6.	REQUIREMENTS TRACEABILITY	6-1
7.	LIST OF ACRONYMS	7-1

LIST OF APPENDICES

Appendix		Page
APPENDIX A	HUMAN COMPUTER INTERFACE FORMS	A-1
APPENDIX B	DIINS ADM SOFTWARE OPERATION	B-1
APPENDIX C	DIINS HCI CONTROL AND DISPLAY SOFTWARE DESIGN	C-1
APPENDIX D	DATA LOGGING REQUIREMENTS	D-1
APPENDIX E	NEW SHINNADS TO DIINS INTERFACE REQUIREMENTS	E-1
APPENDIX F	DIINS ADM SENSOR CONFIGURATION DATA ORGANIZATION	F-1
APPENDIX G	DIINS ADM SENSOR DATA PROCESSOR SOFTWARE DESIGN	G-1
APPENDIX H	DIINS LIBRARY, SYSTEM AND UTILITY DIRECTORY CONTENT	H-1
APPENDIX I	'WITHED' PACKAGES FOR DATA COLLECTION TASKS	I-1

LIST OF FIGURES

Figure — Page

Figure 4.2-1 DIINS ADM Software Tasking Architecture 4-14
Figure 4.2-2 Task Master Flow Chart 4-23
Figure 4.2-3 Mother of All Interrupt Handlers Flow Chart 4-24
Figure 4.2-4 Port Handlers and Interrupt Failure Handler Flow Chart 4-25
Figure 4.2-5 Device Drivers Flow Chart 4-26
Figure 4.2-6 Real Sensor: Internav LC360 (Loran-C) Flow Chart 4-27
Figure 4.2-7 Real Sensor: Sperry MK29 INS Flow Chart 4-28
Figure 4.2-8 Real Sensor: Trimble TANS PY (GPS) Flow Chart 4-29
Figure 4.2-9 Simulated Sensors Flow Chart 4-30
Figure 4.2-10 Prefilter/Filter Flow Chart 4-31
Figure 4.2-11 Prefilter/Filter Flow Chart (Cont'd) 4-32
Figure 4.2-12 Prefilter/Filter Flow Chart (Cont'd) 4-33
Figure 4.2-13 Additional Secondary Factors Flow Chart 4-34
Figure 4.2-14 Failure Detection and Isolation Flow Chart 4-35
Figure 4.2-15 Data Logger Flow Chart 4-36
Figure 4.2-16 Data Display Flow Chart 4-37
Figure 5.1-1 DIINS Overall Ada "Withing" Diagram 5-2
Figure 5.2-1 Display Data Task (DDT) Ada Unit Dependency Diagram 5-3
Figure 5.6-1 Data Collector CSC Design 5-5
Figure 5.6-2 Data Logger Task (DLT) Ada Unit Dependency Diagram 5-8
Figure 5.7-1 Sensor Data Collection Tasks (DCT) Ada Unit Dependency Diagram 5-10
Figure 5.7-2 Sensor CSC Design 5-12
Figure 5.9-1 Unifilter Ada Unit Dependency Diagram 5-15
Figure 5.9-2 Additional Secondary Factors (ASF) Ada Unit Dependency Diagram 5-16
Figure 5.9-3 High Rate Attitude Data Distribution Sequence Diagram 5-17
Figure 5.10-1 Failure Detection and Isolation (FDI) Ada Unit Diagram 5-18
Figure A-1 Top Level DIINS HCI Form A-5
Figure A-2 Sensor Selection Form (from installed sensors) A-6
Figure A-3 DIINS Maintenance Mode Menu Selections A-7
Figure A-4 Sensor Installation Form A-8
Figure A-5 Sensor Lever Arm Setup Form A-9
Figure A-6 Sensor Communication Channel Setup Form A-10
Figure A-7 FDI Parameters Setup Form A-11
Figure A-8 Data Logging Setup Form A-12
Figure A-9 Failure Injection Form A-13
Figure A-10 System Status Form A-14
Figure B.2-1 DIINS ADM Use Case Model B-2
Figure B.3-1 DIINS ADM Initialization B-3
Figure B.4-1 DIINS ADM Sensor Configuration Data Setup B-4
Figure B.5-1 DIINS ADM Operation B-5

Figure | Page

Figure C.2-1 HCI Control And Display Package C-1
Figure C.2-2 HCI Control And Display CSC Sub-Packages C-2
Figure C.2-3 Software Component Packages C-2
Figure C.3.1-1 Forms Package Classes C-3
Figure C.3.2-1 Sensors Package Classes C-4
Figure C.3.3-1 Filters Package Classes C-4
Figure C.3.4-1 Failure Parameters Package Classes C-5
Figure C.3.5-1 Communications Package Classes C-5
Figure C.3.6-1 System Package Classes C-5
Figure C.4.1-1 Sensor Installation Sequence Diagram C-6
Figure C.4.2-1 Select Sensors Sequence Diagram C-7
Figure C.4.3-1 Lever Arm Setup Sequence Diagram C-8
Figure C.4.4-1 Communications Parameters Setup Sequence Diagram C-9
Figure C.4.5-1 FDI Parameters Setup Sequence Diagram C-10
Figure C.4.6-1 Data Logging Setup Sequence Diagram C-11
Figure C.4.7-1 Failure Injection Sequence Diagram C-12
Figure C.4.8-1 System Status Monitor Sequence Diagram C-13
Figure G.2-1 DIINS HCI Control And Display Ada Code G-1
Figure G.2-2 Ada Interface Software Packages G-2
Figure G.3.1-1 Data Collector Class G-2
Figure G.3.2-1 CPU Usage Class G-3
Figure G.3.3-1 FDI Class G-3
Figure G.3.4-1 Prefilter/Filter Class G-3
Figure G.3.5-1 Other Software Component Class G-4
Figure G.3.6-1 FOPVAL Data Class G-4
Figure G.3.7-1 SimSetupData Class G-5
Figure G.3.8-1 Communications Classes G-6
Figure G.4.1-1 Sensor Installation Sequence G-7
Figure G.4.2-1 Select Sensors Sequence Diagram G-8
Figure G.4.3-1 Lever Arm Setup Sequence Diagram G-9
Figure G.4.4-1 Communications Parameters Setup Sequence Diagram G-10
Figure G.4.5-1 FDI Parameters Setup Sequence Diagram G-11
Figure G.4.6-1 Data Logging Setup Sequence Diagram G-12
Figure G.4.7-1 Failure Injection Sequence Diagram G-13
Figure G.4.8-1 System Status Monitor Sequence Diagram G-14

LIST OF TABLES

Table — Page

Table 4.1-1 HCI, Control and Display CSC Requirements 4-5
Table 4.1-2 Device Drivers Requirements 4-8
Table 4.1-3 Encoders CSC Requirements 4-9
Table 4.1-4 Data Collector CSC Requirements 4-10
Table 4.1-5 Sensor CSC Requirements 4-11
Table 4.1-6 Bounded Buffer CSC Requirements 4-11
Table 4.1-7 Prefilter/Filter CSC Requirements 4-12
Table 4.1-8 FDI CSC Requirements 4-12
Table 4.1-9 Task Master CSC Requirements 4-12
Table 4.1-10 System Configuration Controller CSC Requirements 4-13

1. SCOPE

1.1 Identification

This Software Design Document (SDD) specifies the high level and detailed design of the software components and interfaces associated with the Computer Software Configuration Item (CSCI) for the Dual Inertial Integrated Navigation System (DIINS) Advanced Development Model (ADM) program. The design described in the SDD satisfies the requirements in the DIINS Request For Proposal, Reference [2-1] in line with the PRIOR proposal in response to the RFP, Reference [2-2] , and as defined in detail in the DIINS System Specification, Reference [2-3]. In addition, the design is derived from the design of the DIINS eXperimental Development Model (XDM) which was developed by the staff of the Defence Research Establishment Ottawa, and has previously undertaken successful sea trials. The system design of the DIINS ADM was investigated previously during DIINS ADM Phase 1 activities and the results of these investigations, including the definition of the ADM software architecture, were documented in the Phase 1 Report, Reference [2-4].

1.2 System Overview

The following project background information is reproduced from Reference [2-1].

A need for an integrated navigation system for the Canadian Navy's major warships has existed for some time. Present navigation instruments on the ships operate independently and significant operator effort is required to monitor the navigation suite and determine the best estimate of the ship's true position. The Defence Research Establishment Ottawa (DREO) has conducted an extensive research and development program into the development of a DIINS system that integrates the data from the various on-board navigation sensors and automatically computes the best estimate of the ship's true position, speed, heading and attitude. At the same time the system monitors the health of the various sensors to determine even very subtle sensor abnormalities, and can reconfigure the sensor suite so that the navigation solution is not corrupted by faulty sensor data.

The DIINS developed at DREO is considered an experimental development model (XDM) but it has undergone extensive laboratory testing, dedicated sea trials aboard a research vessel, and an operational evaluation on one of the Navy's Halifax class frigates, HMCS Calgary. The current DIINS XDM software consists of approximately 30,000 lines of Ada code, and the real-time hardware consists of 3 VME bus computer cards and associated components. A separate effort has led to the development of SHINNADS - the Shipboard Integrated Navigation and Display System. SHINNADS - which is being built by Offshore Systems Ltd. of Vancouver B.C. - is an ECDIS (Electronic Chart and Display Information System) that has been specially modified to operate in the navy environment. Now, an Advanced Development Model (ADM) of DIINS is to be designed and built to be inserted into the SHINNADS console in order to provide highly accurate and reliable navigation data to drive the electronic charts of SHINNADS.

1.3 Document Overview

Section 4 of this document presents a high level design description of the DIINS ADM software. Section 5 presents a more detailed description of the design of each Computer Software Component (CSC) constituting the DIINS ADM CSCI. Detailed design information has also been included in a number of appendices. In particular, Appendix A presents the details of the Human Computer Interface (HCI) associated with the ADM to allow the maintainer to configure DIINS correctly for operation, while Appendices B and C show the detailed behavior of the software in response to actions that take place via the HCI.

1.4 Acknowledgments

PRIOR would like to acknowledge the support given during the system and software design analysis by the DREO DIINS staff, in particular Chrisopher McMillan, Jeff Bird, Lum Hwang and Dale Arden.

PRIOR would also like to acknowledge the contributions made during the system and software design analysis by Harvey Russel of Offshore Systems and Bruno Scherzinger of Applanix.

2. REFERENCED DOCUMENTS

[2-1]	Request For Proposal Dual Intertial Integ Navigation	Solicitation No. W7714-6-0043 A OBS Ref No. PW-SSSV-041-346
[2-2]	Management & Technical Proposal Dual Inertial Integrated Navigation System	PRIOR Doc. No. K21049
[2-3]	DIINS ADM Development System Specification	PRIOR Doc. No. 98-XXXX-K
[2-4]	DIINS ADM Development Phase 1 Report: System Design Review	PRIOR Doc. 97-0119-K
[2-5]	Techniques for Soft-Failure Detection in a Multi-sensor Integrated System	J. Christopher McMillan, Jeffrey. S Bird Dale A.G.Arden, Navigation - Journal Of The Institute Of Navigation, Vol. 40, No. 3, Fall 1993
[2-6]	DIINS Software Enhancement Study: Final Report	Contract W8462-2-FJ0A/02-ER Task: SESS-010, 29 March 1996 PRIOR Doc. 95-0171
[2-7]	Kalman Filter Design Of Dual Inertial Integrated Navigation System (U)	D.A.G. Arden Dale Arden Consulting, Ottawa Report No. DAC-02-97 August 1997
[2-8]	DIINS Failure Detection Isolation And Reconfiguration Procedures (U)	D.A.G. Arden Dale Arden Consulting, Ottawa Report No. DAC-03-97 August 1997
[2-9]	Analysis Of Real Time Results From DIINS 1995 Sea Trials (U)	D.A.G. Arden Dale Arden Consulting, Ottawa Report No. DAC-01-97 August 1997
[2-10]	NMEA 0183 Standard for Interfacing Marine Electronic Devices	National Marine Electronics Association Version 2.00, January 1, 1992.

[2-11] Air Standardization Coordinating Committee: "The Specification for Evaluation Of The Accuracy Of Hybrid Navigation Systems" — ASCC AIR STD 53/16A, 6 April 1988

3. CSCI-WIDE DESIGN DECISIONS

Figure 3.1 shows the interfaces involved for the ADM. The DIINS CSCI software is shown in the center of the diagram, with the equipment that it interfaces to being shown around the periphery of the diagram.

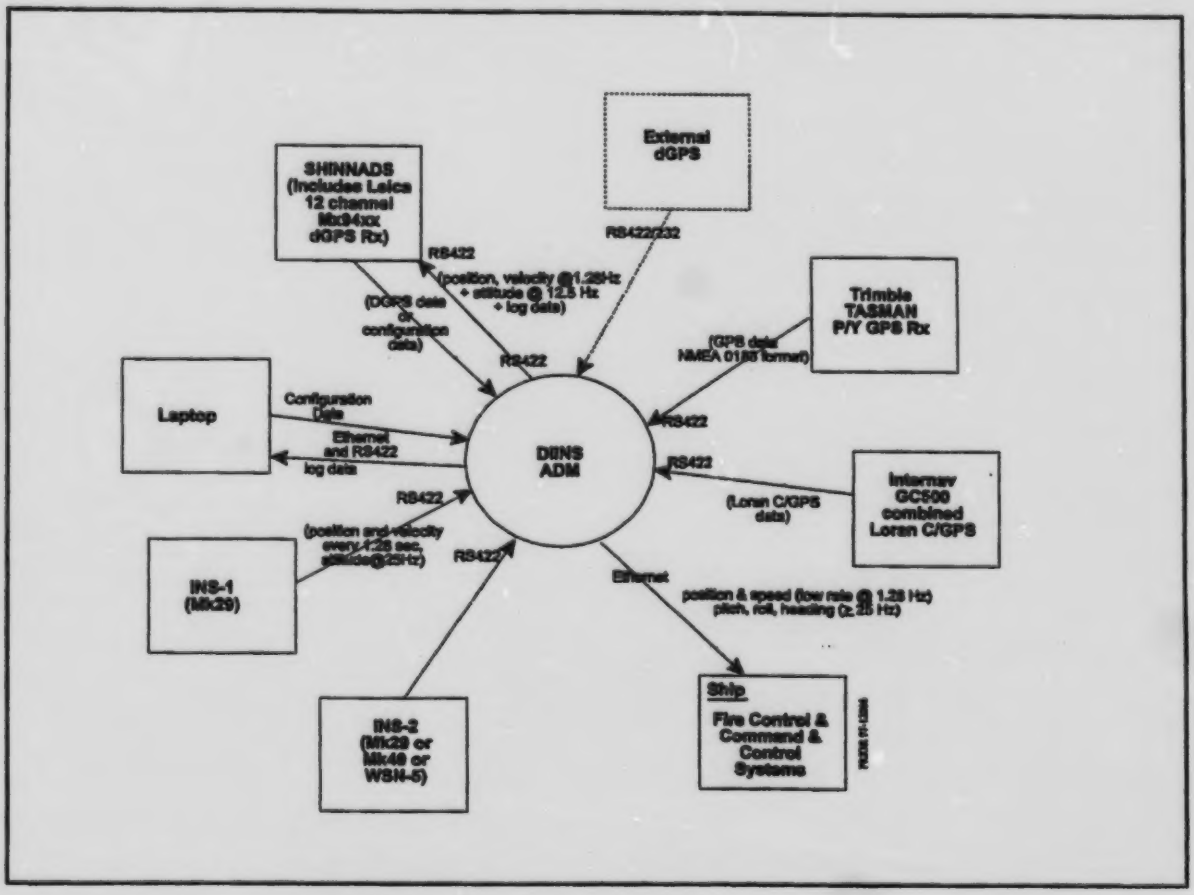


Figure 3-1 DIINS ADM Context Diagram

4. CSCI ARCHITECTURAL DESIGN

4.1 CSCI Components

The following sections indicate the high level architectural design of the DIINS ADM software, and the allocation of the DIINS ADM system requirements to the various CSCs within the DIINS ADM CSCI. In general the requirements have been allocated to a single component, the main component fulfilling the requirement. In some cases, in practice, a number of CSCs have to work together to fulfill the overall requirement. This interaction/cooperation between the components will be factored into the high level and detailed design presented in this document. The repetition of requirements in more than one place within the document has been avoided mainly to improve the maintainability of the document.

4.1.1 High Level Architecture

Figure 4-1 illustrates the high level system/software design for a proposed DIINS ADM architecture to be used in the operational environment. The following sections describe the main components of the software and highlight changes from the current implementation that make the architecture more generic and easily adaptable to new input/output requirements.

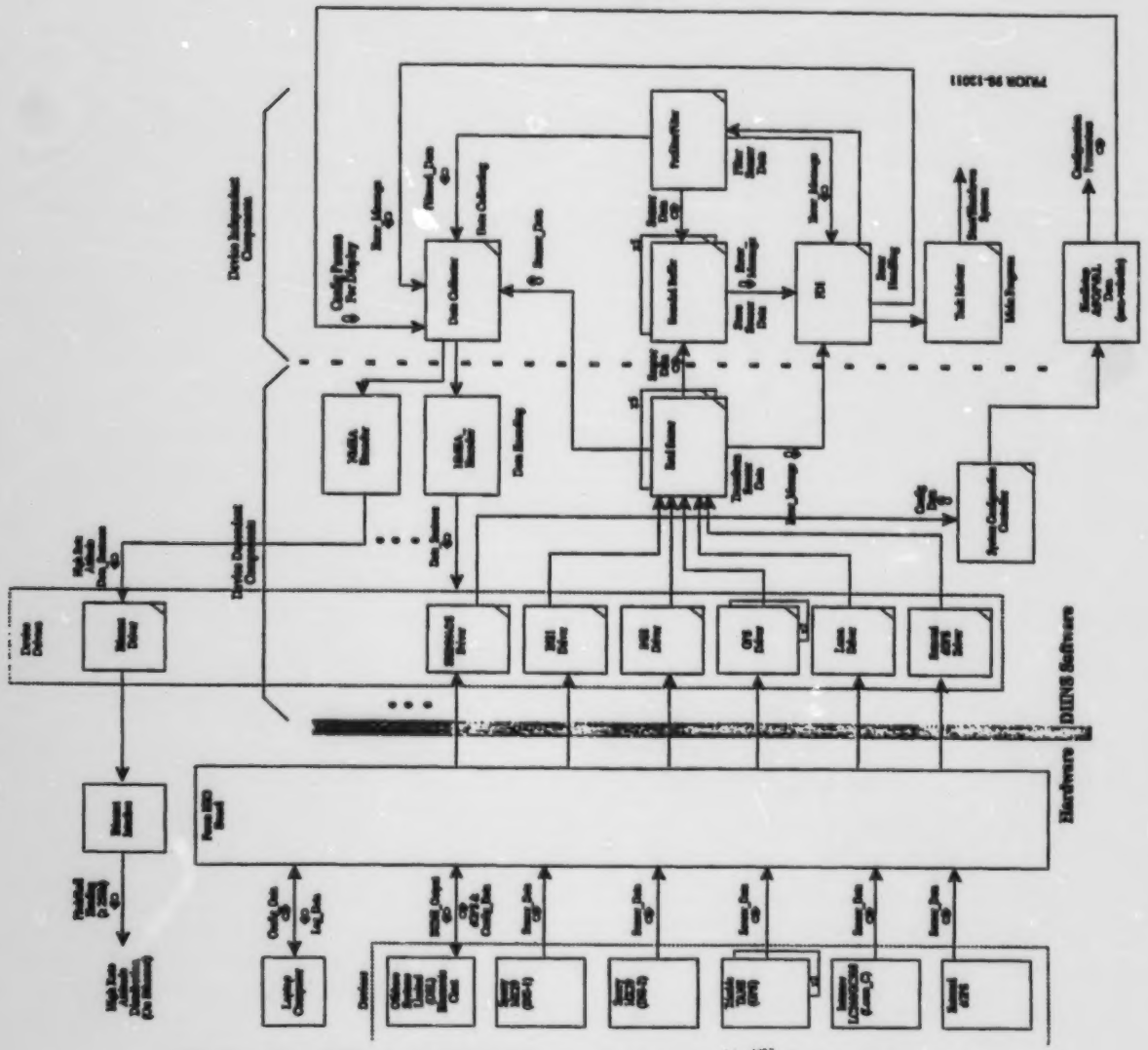


Figure 4-1 DIINS ADM Software Architecture High Level Design

4.1.1.1 Device Drivers

The OSL and sensor hardware interface to the DIINS software is via a ForceISIO VME RS232/RS422 interface board. A Device Driver component will be associated with each hardware device. Port and interrupt handling would be encapsulated within the Device Driver. Adding hardware devices to DIINS or changing interface boards will require the development of additional drivers or modification of existing drivers. The OSL driver outputs DIINS data to the OSL Electronic Chart for display. The sensor drivers read data from the sensors, via the ForceISIO board, and send the data on to the Real Sensor component associated with the driver.

4.1.1.2 Real Sensor

In the current implementation of DIINS, the Real Sensor software components (as opposed to the Simulated Sensors which are also available within the software to support the development activities) are similar in structure and perform similar functions. The Real Sensor components in the enhanced design will be instantiations of a generic package. A Real Sensor package will be instantiated for each sensor connected to the DIINS software. The generic package will be described in more detail in **Section 5.7**. The Real Sensor component will receive the raw sensor data from its associated device driver. The raw sensor data will be transformed into a record structure containing the sensor readings (e.g., Latitude, Longitude, Heading, etc.). The sensor readings will then be passed to the Data Collector and Bounded Buffer components of the system.

4.1.1.3 Bounded Buffer

The Bounded Buffer components of the system store the sensor readings received from the Real Sensors. As in the current implementation of DIINS, the Bounded Buffers will be instantiations of the generic queue package. A Bounded Buffer package will be instantiated for each sensor connected to the DIINS software.

4.1.1.4 Prefilter/Filter

The Prefilter/Filter reads the sensor data from the Bounded Buffer associated with each sensor. Filtered data is then passed to the Data Collector component. The Additional Secondary Factors routines in the current implementation of DIINS will be encapsulated in the Prefilter/Filter component in the enhanced design of DIINS. In the current developmental implementation of DIINS, the number of filters operated and their configurations, in terms of the sensor data included in the filters, is defined in a configuration file (refer to Ref[2-4], Section 4). In the enhanced operational design of DIINS, this default configuration information will be stored in non-volatile storage, with the parameters being able to be displayed and modified via the user interface. The actual access to the configuration data within the system will be via a collection of functions which the System Configuration Controller can use to set or retrieve the configuration information. The use of functions allows the configuration parameters to be hidden within the body of a package, thus avoiding recompilation of the whole system when a default parameter is changed. The enhanced Prefilter/Filter design will be implemented so that only the configuration information need be modified if an additional sensor is connected to the DIINS software. See Appendix F for details of the DIINS ADM sensor configuration data design requirements.

4.1.1.5 FDI

In the current implementation of DIINS, the FDI performs both hard and soft sensor failure detection and isolation. In addition to this capability, the enhanced FDI will handle software exceptions. When an exception is raised in one of the software components in the system, an error message will be generated identifying the exception and the source of the exception. The error will also be flagged as critical or non-critical. As an example, a Tasking Error exception raised in one of the system tasks would be considered a critical error but a Constraint Error exception on a sensor value would not. The error message will then be passed to the FDI. If the error is critical, the FDI will call the Task Master to initiate a shutdown of the system. An error message indicating the nature of the error, its location, and its criticality will be passed to the Data Collector component for display on the output interface. This ability to display a software error and the source of the error will support the maintenance of the DIINS software during operation.

4.1.1.6 Data Collector

The Data Collector component of the system receives sensor data from the Real Sensors, filtered data from the Prefilter/Filter, and error messages from the FDI. This information is then passed to the NMEA Encoder CSC for encoding to be sent to the OSL Electronic Chart for display and to the portable PC display interface for system maintenance purposes. The data is also logged on a mass storage device, e.g. disk or tape. A more detailed design description of the Data Collector component is contained in **Section 5.6**.

4.1.1.7 NMEA Encoder

The NMEA Encoder component of the system would receive sensor data and error messages from the Data Collector component. This information is then formatted into data sentences to be passed through the OSL device driver to the OSL Electronic Chart for display. Changes to the output format or additional data sentences will require modifications to existing routines or additional formatting routines within the NMEA Encoder component. A more detailed design description of the enhanced NMEA Encoder component is described in **Section 5.5**.

4.1.1.8 Task Master

The Task Master component controls the overall operation of the DIINS software. On start up, Task Master will instantiate the appropriate number of generic Real Sensor and Bounded Buffer packages (based on information from the system configuration package) and initiate system start up. Task Master will also perform an orderly shutdown of the system when a critical error is detected (Note: The critical error message would be output before DIINS software is terminated.).

4.1.2 Human Computer Interface, Control and Display CSC

Table 4.1-1 HCI, Control and Display CSC Requirements

a	This Control and Display software shall provide two equivalent methods of accessing the DIINS internal parameters and configuration files: i) via the SHINNADS console and the two way serial communication link to DIINS ii) via a separate hand-held or laptop device attached to an available communications port on DIINS.	3.D
b	The Control and Display software must be able to	4.D

	Summary Of Requirement	System Req. Traceability
	accommodate at least the sensors defined in 4.1.2 (e).	
c	The navigation system to be used shall be selected from a list.	4.E
d	The DIINS software shall enable the user to view and optionally change some of the DIINS configuration and filter parameters that are presently set in the DIINS configuration and filter parameter files (SimSetup.cfg and FilterInitx.dat). The parameter values currently in the DIINS XDM parameters files that are not defined below will be setup to appropriate values by the system during initialization. The parameters accessible to the ADM user will be as follows: <u>Configuration Parameters:</u> • Definition of the installed sensor models in DIINS; • Selection of the sensors to be used with DIINS; • Lever Arms for the installed sensors; • RS-232/422 communication channel parameters for the installed sensors; Initial sensor measurement values (for filter initialization), including standard deviations and failure and pass fractions and times.	12.B
e	There shall be two essentially equivalent ways to view and change parameters: a. via the User Interface software in the hand held of laptop computer (external control and display unit) interfaced to DIINS through an unused serial port; and b. via a custom screen software on the SHINNADS console.	12.C
f	The control and display functions for the DIINS internal parameters, when presented on the SHINNADS console screen shall be consistent with other control and display screens used in SHINNADS.	12.D

	Summary Of Requirement	System Ref Trace Ellis
g	The external control and display unit software shall present a similar interface as used when accessing DIINS from the SHINNADS console.	12.E
h	The software shall allow the operator to specify the source (if any) of DGPS data as one of the following two sources: a. SHINNADS DGPS Receiver b. Separate DGPS system	12.F
i	The software shall allow the operator to select the INS , GPS and other navigation sensors/systems to be used from a list of the available sensors/systems.	12.G
j	The sensor and Kalman filter configuration interface shall be sufficiently flexible to allow a knowledgeable user the ability to fine tune the parameters of a particular sensor.	12.H
k	The DIINS ADM software (via the SHINNADS display) shall provide the following outputs when operating in the Maintenance Mode: • The Percentage used of the available processing power • The filter currently selected for output • Identified software exceptions • FDI Log text messages • Filter Log text messages	12.I
l	When operating in Maintenance Mode the software shall allow the operator to inject a hard or soft sensor failure for a selected sensor, which will cause the Sensor CSC to degrade the data accordingly. This will be used to verify the correct operation (including failure detection performance) of the DIINS failure detection processing.	12.J
m	The operator shall have to take some specific action to allow the configuration control HCI forms to be presented.	12.M

4.1.3 Device Drivers CSC

Table 4.1-2 Device Drivers Requirements

	Summary of Requirement	System Requirements
a	Communication between the DIINS processor and the SHINNADS processor will be via a link internal to the console.	2.C
b	In the ADM, this link shall be a two way RS-422 link.	2.D
c	The software shall be capable of reading both RS232 and RS422 serial data from the various sensors.	3.C
d	The data distribution rate will be ...	3.E
e	The software shall have the ability to interface to at least the following sensors: Either: - Dual Sperry Mk-29 Mod 3 Inertial Navigation Systems or - (pending DND procurement approval for the MK-49) - Dual Sperry Mk-49 (NATO SINS) Inertial Navigation System and - Trimble TASMAN (P/Y) GPS receiver. - Leica 12 channel MX94xx series DGPS receivers - Internav GC500 combined Loran C / GPS(C/A) receiver Or the - Internav LC360 Loran C receiver And - Differential GPS (DGPS), from a system external to SHINNADS	4.A
f	The allowable DGPS inputs shall be as follows: i. via a modified link with CSL's SHINNADS to allow the DGPS receiver within SHINNADS to provide DGPS data to DIINS; ii. via a separate DGPS system that can supply DGPS data in NMEA formats, through an unused	4.B

	Summary Of Requirement	System Req. Traceability
	serial port on the DIINS ADM.	
g	The best integrated estimate of vessel position, speed and heading shall be output on an RS-422 serial line to SHINNADS at least once every 1.28 seconds.	7.A
h	The best estimate of vessel roll, pitch and heading shall be output at least every 40 milliseconds.	7.B
i	The software shall distribute the high rate roll, pitch and heading data (as well as the lower rate position and speed) from DIINS to the rest of the ship. There are 3 distribution requirements: a) Pos/speed/heading and internal DIINS diagnostic info to SHINNADS at ~ 1/second; (Currently met by XDM) b) Pos/speed/heading at ~1/sec for general shipboard system uses; and (Currently met by XDM) c) High rate (25 Hz or greater), low latency (less than or equal to 40 ms) Roll/Pitch/Heading data for weapons or other systems requiring precise, near-continuous attitude information. (New for ADM)	7.C

4.1.4 Encoders CSC

Table 4.1-3 Encoders CSC Requirements

	Summary Of Requirement	System Req. Traceability
a	The Encoder software must be able to accommodate at least the sensors defined in 4.1.2 (e).	4.D

4.1.5 Data Collector CSC

Table 4.1-4 Data Collector CSC Requirements

	Summary of Requirement	Associated Traceability
a	The software shall be capable of logging DIINS navigation outputs and other, user-selectable Kalman filter outputs, at rates and durations specified by an operator using either of the control and display methods above. The data to be logged shall be as follows: Raw Sensor Data: • INS • GPS & DGPS • Loran Kalman Filter Outputs: • Filter Residuals • State Vector • Position Estimate • Covariances Miscellaneous • Identified software exceptions • FDI Log text messages • Filter Log text messages	3.F
b	The maximum volume of data that can be logged without changing the removable hardware media shall be: • 1GB on Portable PC hard drive • TBD (4 or 8GB) on portable EXABYTE tape drive.	3.F
c	The data shall be logged with the format defined in Appendix TBD (compression may be used).	3.H
d	The DIINS data shall be logged in a sequence of log files, each file covering a two hour time period, with no data gaps between files.	3.I

	Summary Of Requirement	System Req Traceability
e	The names of the log files shall be automatically generated by the system, and will be based on a function of DIINS time.	3.J
f	The logged data shall be able to be replayed at an on-shore analysis facility.	3.K

4.1.6 Sensor CSC

Table 4.1-5 Sensor CSC Requirements

a	The Sensor software must be able to accommodate at least the sensors defined in 4.1.2 (e).	4.D
b	The software shall contain a parameterized sensor library, in which a few of the most common and likely naval INS (specifically the Sperry MK-49 and Litton WSN-5 (the INS used on the TRUMP Class ships)), GPS and other navigation systems can be selected from a list.	4.E
c	Minimal software changes shall be required to interface to a new sensor. Thus the areas of the code that must be changed to accommodate a new sensor must be kept as separate as possible and documented thoroughly.	4.F
d	When operating in Maintenance Mode the software shall inject a hard or soft sensor failure for a selected sensor, in response to requests for failure injection from the Human Computer Interface software.	12.J

4.1.7 Bounded Buffer CSC

Table 4.1-6 Bounded Buffer CSC Requirements

.	Satisfies no explicit System Requirements.	

4.1.8 Prefilter/Filter CSC

Table 4.1-7 Prefilter/Filter CSC Requirements

a	The Prefilter/Filter software must be able to accommodate at least the sensors defined in 4.1.2 (e).	4.D
b	The software shall be capable of providing navigational accuracies equivalent to those obtained by the XDM, approximately 15-20m (95%) N and E position error and 1-2 arcmin (rms) roll pitch and heading error (Mk-29 integrated with Tasman P/Y GPS).	5.A

4.1.9 Failure Detection and Isolation Controller CSC

Table 4.1-8 FDI CSC Requirements

a	The FDI software must be able to accommodate at least the sensors defined in 4.1.2 (e).	4.D
b	The software shall implement FDIR algorithms equivalent to the Kalman filter residual testing, chi-squared fault detection and multi-filter failure isolation techniques within the DIINS XDM.	6.A
c	In addition the methodology, as presented in the XDM, of filter selection and reconfiguration under sensor failure conditions, shall be followed.	6.B

4.1.10 Task Master CSC

Table 4.1-9 Task Master CSC Requirements

	Satisfies no explicit System Requirements.	

4.1.11 System Configuration Controller CSC

Table 4.1-10 System Configuration Controller CSC Requirements

	Summary of Requirement	System Req. Traceability
a	The DIINS software shall be able to be started without the need to use the HCI to configure the system.	12.K
b	Once the system has been installed and configured, this configuration shall be used as the default configuration.	12.L

4.2 Concept of Execution

This section presents the details of the operation and behavior of the DIINS ADM software. In addition, Appendix B presents some Sequence Diagrams of the main data flows in the DIINS software. These diagrams were produced using the Rational/Rose software modeling tool.

4.2.1 DIINS Tasking Architecture

Figure 4.2-1 illustrates the overall tasking architecture of the DIINS software. The following sections describe the overall operation of the system based on this tasking architecture.

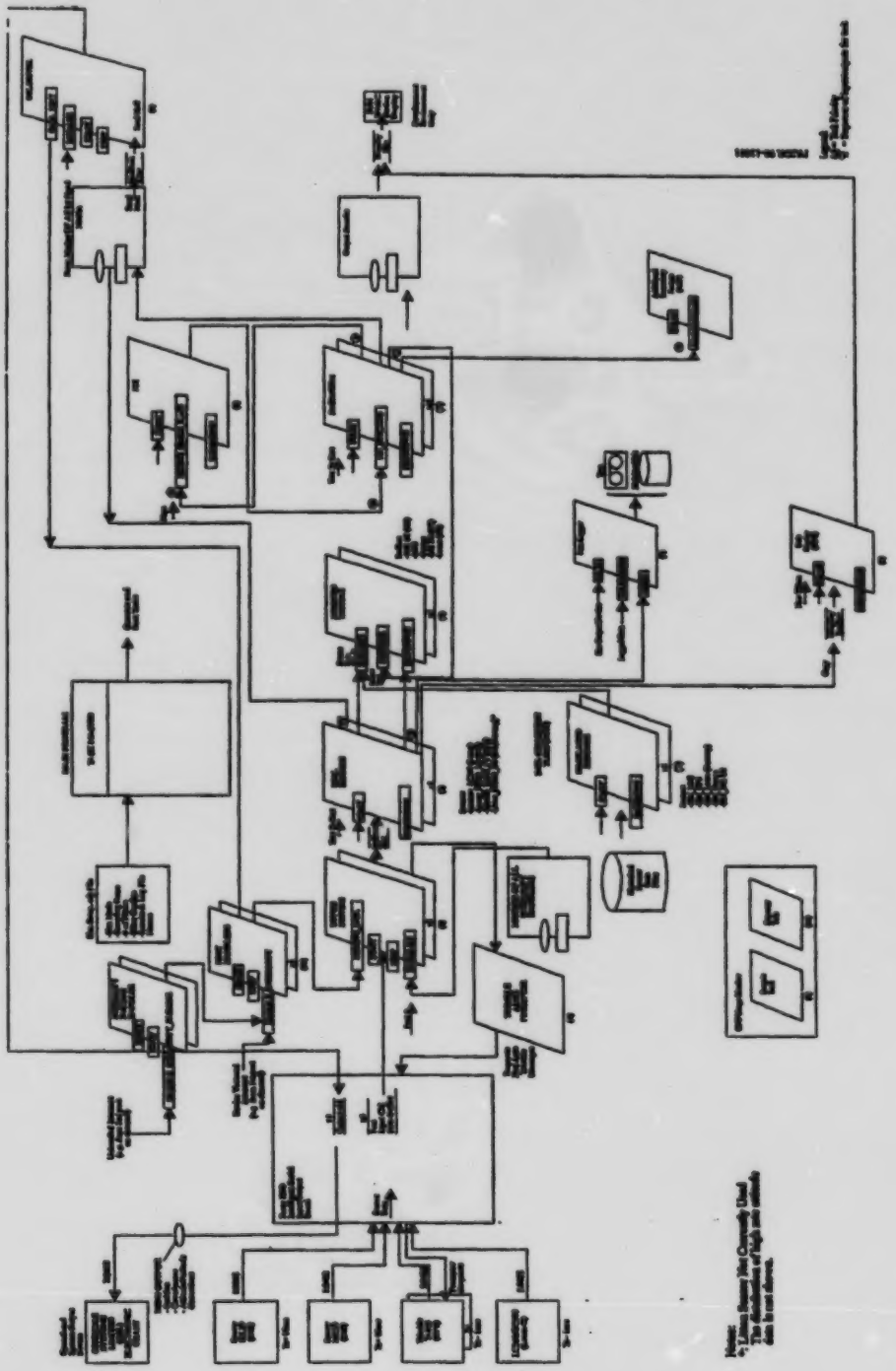


Figure 4.2-1 DIINS ADM Software Tasking Architecture

4.2.1.1 Task Master

The procedure TaskMaster controls the overall operation of the DIINS software. TaskMaster uses the system configuration parameters to set up the overall configuration of the simulation. TaskMaster creates and starts all of the other tasks associated with the system, including the device drivers associated with the input/output interfaces to the sensors. TaskMaster assigns the sensors to the various input/output ports on the ForceISIO interface based on the configuration data settings. The configuration data includes all of the information required to configure the simulation for operation, apart from the filter parameters, which are maintained in a separate file. Parameters contained in the configuration file include:

- Simulation mode selection (REAL_TIME, SIMULATION).
- Log file size, i.e. whether a single (BIG_FILE) is used for logging, or whether separate files are used to log data from the FDI and filter processing. An output medium parameter also specifies on which device to store the logged data (e.g. LOCAL_DISK, NFS_DISK, SHADOW_TAPES).
- Sensor sampling times.
- The number of filters used and their configurations (i.e. sensors used in each filter).
- The simulation duration to be used in SIMULATION mode.
- The names of the files to be used for displaying the results of the simulation,
- The names of the files to be used for data logging.

4.2.1.2 Real Sensors

The real sensor hardware is interfaced to the DIINS ADM software via a ForceISIO VME RS232/RS422 interface board. The Force interface contains the Channel Buffers used to buffer the data to and from the sensors. The sensors input their data at their individual sampling rates, which are defined in the configuration data. For nominal system operation the system will be configured to input the Sperry MK29 INS data every 40ms, and the GPS and Loran data every second. When the data is available from a sensor it activates the port handler via a vectored program interrupt, and the port handler notifies the device driver for the sensor to RECEIVE_DATA. The device driver reads the data from the channel memory and deposits the data in a Mailbox associated with the device. The data is transferred from the sensors to the DIINS environment in blocks, e.g. 33 byte blocks for the Sperry MK29 sensor. The DIINS Mail System functions as shared memory between the tasks, and operates more efficiently than transferring the data between the two tasks via Ada rendezvous. The Mailsystem is a typical FIFO task, as found in most real-time executives.

As discussed above, all data input from the sensors is controlled by the sensor, with the data input being notified via a program interrupt. Control of the sensor operation is achieved via program-controlled output to the sensors via the ForceISIO interface board, and is not interrupt driven. The Trimble GPS sensor (outputting Trimble's native TSIP/TIPY) provides automatically a number of messages as input data via the ForceISIO port. Additional, non-automatic, inputs can be requested by the system using the Trimble Auto Prompter task, the

operation of which is triggered by the device driver, and results in data requests being sent to the Trimble via the ForceISIO interface. Examples of additional data that can be requested by the Auto Prompter are as follows:

- Trimble software version
- GPS System Message
- Tracking Status
- Measurement Correction

The Trimble GPS sensor outputting NMEA formatted data (as opposed to the one outputting the Trimble proprietary format, TSIP/TIPY) is not able to receive any commands, and thus also requires the Auto Prompter task.

The Real Sensor tasks shown in **Figure 4.2-1** read the sensor data from the Mailbox and then transform the data from the mail transfer format into the internal data structures used during the sensor data calculations. Following transformation the data is placed into the Bounded Buffer associated with the sensor, in order that it can be retrieved by the Prefilter/Filter tasks. Sensor data is sent to the Data Logger task in order that it can be logged. The sensor data received is time stamped with its time of arrival, and the time period between the new data and the last data received is monitored. If the period exceeds one and a half times the sensor sampling period sensor data is assumed to have been lost. When data is lost the last data received is used to 'fill' the data input stream, and the loss of the input data in indicated in the status associated with the data. Timing in the DIINS software is performed using the Calendar.Clock function. The system clock is initialized at the start of a test run by using GPS time available from the Trimble GPS sensors (designated by UseMyClock parameter in the SimSetup.cfg file).

4.2.1.3 Simulated Sensors

The simulated sensors (shown on **Figure 4-2-1**, available in the DIINS ADM development environment only) read either computer generated sensor data or real data generated and recorded during sea trials. The data is read from binary files stored on the disks associated with the Network File Server in the development environment. Binary data storage is used to promote efficient storage of the large volumes of data required during DIINS simulations, which typically cover a period of a day. The simulated sensors are used when the DIINS system is operating in the SIMULATION mode, with the simulation being performed on the host Sun SPARC station or the VME-based target.

When the simulated sensors are operational, their data is input to the filters via the Bounded Buffers, as is the case for the real sensor input data.

4.2.1.4 Prefilter/Filter

4.2.1.4.1 Low Rate Data Prefilter/Filter

See Reference[2-7] for details of the filtering concepts applied in the DIINS software. The number of filters operated and their configurations, in terms of the sensor data included in the filters, is defined in the SimSetup.cfg file. The actual filter configuration used, i.e. number of filters and the sensors in each filter, will be programmable within the ADM. The current development system is configured to operate with 15 filters, with the sensor configuration of each filter being as follows:

Filter	#1:	INS1 and INS2
	#2:	INS1, GPS1, GPS2, Loran-C
	#3:	INS2, GPS1, GPS2, Loran-C
	#4:	INS1, GPS1, GPS2
	#5:	INS2, GPS1, GPS2
	#6:	INS1, GPS1, Loran-C
	#7:	INS2, GPS1, Loran-C
	#8:	INS1, GPS2, Loran-C
	#9:	INS2, GPS2, Loran-C
	#10:	INS1, GPS1
	#11:	INS2, GPS1
	#12:	INS1, GPS2
	#13:	INS2, GPS2
	#14:	INS1, Loran-C
	#15:	INS2, Loran-C

The filter reads the sensor data from the Bounded Buffer associated with each sensor. If the Loran-C sensor input is enabled, the filter obtains the Additional Secondary Factors (ASFs) from the ASF task. These factors (in microseconds) are used to compensate for the Loran-C pulse lag caused by the land-path. The factors used are based on data from the Canadian Hydrographic service. The ASF factors are applied every 300 seconds (5 minutes).

The Kalman filter states will be propagated with a period of three times the INS data collection period, which is 1.28 seconds, leading to new filter outputs every 3.84 seconds. The data is actually collected from the INS in 33 byte blocks, one block being sampled every 40 ms. There are eight 33 byte blocks that comprise the total navigation message. The first 15 bytes of each block contain the attitude data refreshed every 40ms. The remaining bytes are the low rate data, consisting of the latitude, longitude, status, etc. bytes that are refreshed every 1.28 sec. It takes 1.28 seconds to get all of the data blocks of interest for the next filter update. The filter measurement algorithms are executed at the data collection rates of the sensors.

The filter checks for sensor measurement failures. If a measurement has previously failed a check on the renewed health of the sensor is made. If the measurement was previously okay, or if the sensor has been reconfigured, the measurement is checked for failure. The sensor data undergoes

two types of tests, Residual Tests and Chi-Square tests. In the Residual test the magnitudes of the Kalman filter residuals associated with the incoming sensor data are compared with their expected covariances. Not all failures will be detectable through the filter residual monitoring. Any slowly accumulating error, such as unmodelled INS error, would tend to be absorbed in the state estimates of other INS errors, and would show little effect on the residuals. The Chi-Square test uses two solutions of each Kalman filter running in parallel, one with measurements, and the other without (shadow filter). The difference between a filter and its shadow is monitored in the Chi-Square test. The shadow filter provides for a statistically significant reasonableness test that is applied to each filters state estimate, to determine if there is a sensor fault.

Each filter provides the results of its Residual and Chi-Square tests to the FDI task as part of the RECEIVE_SENSOR_FLAGS entry call, and the filter receives from the same task entry call an indication as to whether the output of the filter is to be used for data display and whether the filter is okay. The filter waits for an FDI_SYNCHronisation indication from the FDI task before continuing. The filter turns off all measurements formed using a failed sensor, and turns on measurements associated with a sensor with renewed health.

When it is time to output the results the Navigation Best Estimates are produced by correcting the raw navigation data with the state vector estimates to give the best estimate of the true position, and the results are output to the Display File (in the ADM development environment only). If the filter has been selected for display by the FDI task, the output data is converted to National Maritime Electronics Association (NMEA) format and output to the Offshore Systems Limited (OSL) Electronic Chart via the ForceISIO hardware interface. This will be the DIINS ADM operational system interface. The SHINNADS display of the DIINS data which is provided via the DIINS NMEA sentences is as follows:

- TIME
- DIINS:
  - Latitude
  - Longitude
  - Heading
  - Track angle
  - Ground speed
  - Pitch
  - Roll
  - North position covariance
  - East position covariance
  - Cross covariance
  - Status

- GPS1 (TANS P(Y)):
  - Latitude
  - Longitude
  - Track angle
  - Ground speed
- GPS2 (TANS P(Y)):
  - Latitude
  - Longitude
  - Track angle
  - Ground speed
- INS 1 (MK-29):
  - Latitude
  - Longitude
  - Heading
  - Status
- INS 2 (MK-29):
  - Latitude
  - Longitude
  - Heading
  - Status
- LORAN-C (LC360/GC500):
  - Latitude
  - Longitude
  - Status

The filters will also output data to the Sun SPARC station via the Display File in the development environment. This data includes output states and their standard deviations and output residuals and their standard deviations.

If the system is operating in REAL_TIME mode and the filter log file full time is reached (e.g. every two hours), the filter task closes the existing log file and creates a new one.

4.2.1.4.2 High Rate Attitude Data Filter

In order to distribute the attitude data received from the INS at high update rate to the other ship systems, the data is obtained from the Bounded Buffers and is filtered and extrapolated to the time points required for the high rate data distribution. The high rate data filter obtains from the FDI task an indication as to the best INS, and distributes the data for this INS only. The n data points for the next 40 ms period are passed to the Data Collector which passes the data to the NMEA Encoder to encode all of the data to be distributed over the next 40 ms period The

Ethernet Driver distributes the individual attitude updates at the appropriate times. To achieve the distribution rate required it is assumed that the Ethernet Driver which performs the actual attitude data distribution is a low level routine, ideally executing in the Ethernet interface processor, and is not part of the Ada tasking architecture. The majority of the processing for the attitude distribution is performed at the 25 Hz INS attitude data input rate. The Ada software after receiving the attitude input has (40 ms)/n (e.g. 4 ms if n = 10) to work out the attitude points for the next 40 ms and to supply these to the Ethernet Driver for the next attitude distribution point deadline to be met. Once the points for the next 40 ms are supplied to the Driver, there is no further processing to be performed by the Ada tasks until the next attitude measurement input from the selected INS. The limited time available to perform the processing will limit the complexity of the filtering and data extrapolation algorithm implemented. If a more complex algorithm is required to achieve the attitude distribution accuracy, this will limit the attitude distribution rate. In practice, the time available on the DIINS processor to perform the high rate attitude filtering will be limited by the requirements to perform the other filtering and FDI tasks within the processor.

4.2.1.5 Failure Detection and Isolation

The Failure Detection and Isolation (FDI) task accepts the RECEIVE_SENSOR_FLAGS entry calls from all of the active filter tasks before it performs its FDI tasks. When the FDI task has received all of the data from the filters, it releases the filters to continue their operation by calling their FDI_SYNCHRO entries.

The FDI task selects the filter to be used for navigation data display. The nominal filter selected for navigation display would be the one with:

- INS1 or INS2 enabled
- Uses the best available (healthy) aiding sensor
- Uses the largest number of aiding sensors

The FDI task normally selects the filter with either INS1 or INS2 enabled. Under normal conditions, when both INS's are operating properly, DIINS has no preference as to which filter it uses. The DIINS filter with both INS1 and INS2 enabled is used for failure detection only, and not for navigation purposes.

The FDI task performs both hard (where significant changes (discontinuities) in sensor output occur) and soft (where a slight degradation in its output has occurred) failure isolation. The hard sensor failure isolation involves accumulating the residual test results from all Prefilters/Filters and then testing them against thresholds. The results of the Chi-Square tests from the Prefilters/Filters are used to determine (via the use of a lookup table of results of Chi-Squared test results versus the isolated failure) which sensor (hence filters) are affected by a soft failure.

If a failure is detected the details of the new filter configuration are determined, including the details of the filter to be used for display, and the new filter configuration is communicated to the filters at the next RECEIVE_SENSOR_FLAGS entry call.

If the FDI log file time (e.g. 2 hours) is exceeded the existing log file is closed and a new log file is created.

4.2.1.6 Data Logging

The Data Logger task accepts data from the real sensor tasks and logs the data to Exabyte tape drives. The data can also be logged to disk, either to a local disk attached to the target machine or to a disk managed by the Network File Server. In the operational environment, the DIINS ADM will not have a permanently attached Exabyte tape drive. When data logging is required to be performed this will be achieved by attaching a portable Exabyte drive to the DIINS ADM using the available SCSI interface on the DIINS ADM processor board.

The data for recording by tape can be written twice, with the second write being to a shadow Exabyte tape device.

4.2.1.7 Data Display

The Data Display task is used to display data from the real sensors on the Sun SPARC station display; hence, the data display is only available in the development environment. The display task handles the protocol for transferring the data via the local area network (LAN) to the display console. The data displayed includes the following:

- TrimbleTANS GPS: (for GPS1 & GPS2)
  - Latitude
  - Longitude
  - Heading
- Sperry MK29: (for INS1 and INS2)
  - Latitude
  - Longitude
  - Heading
- Internav LC360:
  - Loran Time Delay

- Filter Data:
  - Latitude
  - Longitude
  - Heading
  - Velocity North
  - Velocity East
  - Latitude Covariance
  - Longitude Covariance
- CPU usage (%)

4.2.1.8 Performance Monitor

The DIINS system includes two tasks (Sampler Task and Reporter Task) which are used to determine the load on the CPU. The Sampler task executes at the lowest priority, Priority 0, and continuously runs when no other tasks can run. The task continuously increments a counter, the value of which is proportional to the amount of processing the task has been able to perform in between the execution of the other operational tasks. The Reporter task operates at the highest priority, Priority 63, and displays at the Sun SPARC station the tally of the counts accumulated by the Sampler task over the last seven seconds. The lower the displayed tally, the more heavily loaded is the DIINS CPU.

4.2.1.9 DIINS System Behavior

The following figures show the behavior of the DIINS software in the form of flow charts. These flowcharts define the behavior of the system in detail at the level of the system components shown in **Figure 4-1** and **Figure 4.2-1**. The flowcharts were drawn based on an examination of the Ada code for the various components, and represent a higher level abstraction of behavior that has been implemented in the software.

The flowcharts are self explanatory (to the extent necessary for this design specification), especially when viewed in conjunction with the overall system diagram given in **Figure 4-1**; hence, the details of the flowcharts will not be discussed further.

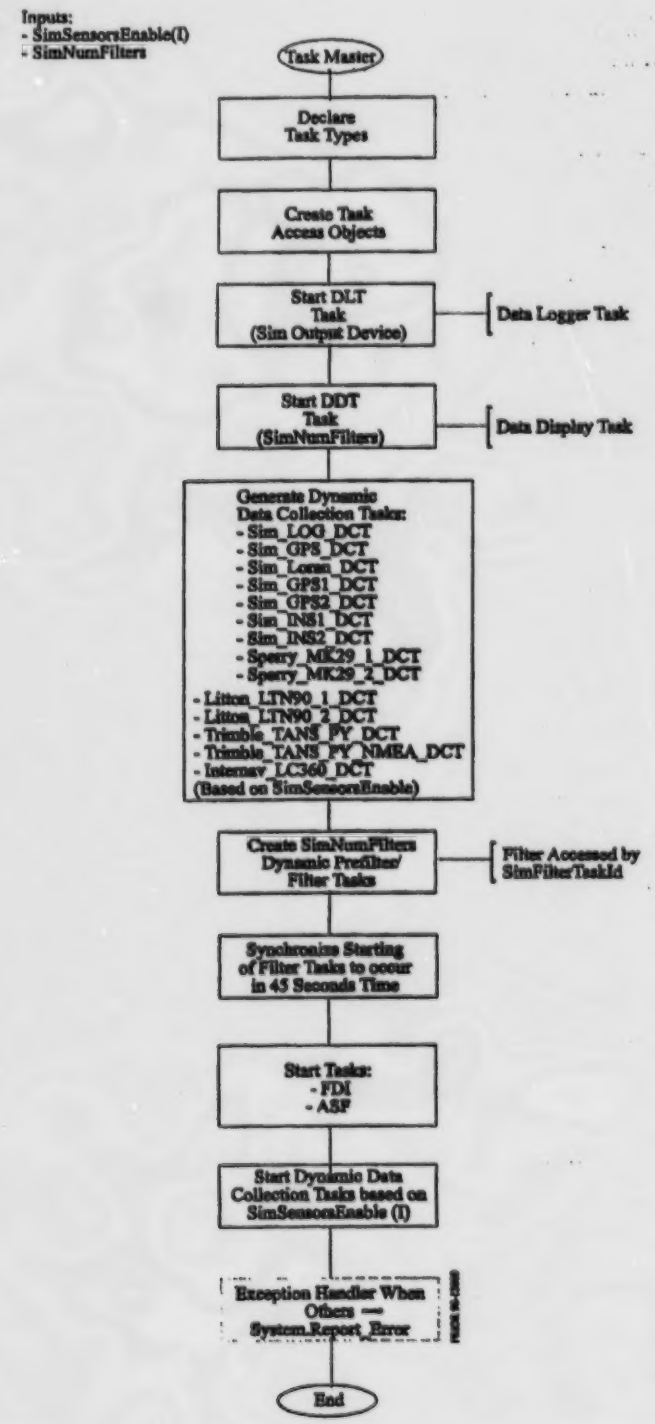


Figure 4.2-2 Task Master Flow Chart

+4

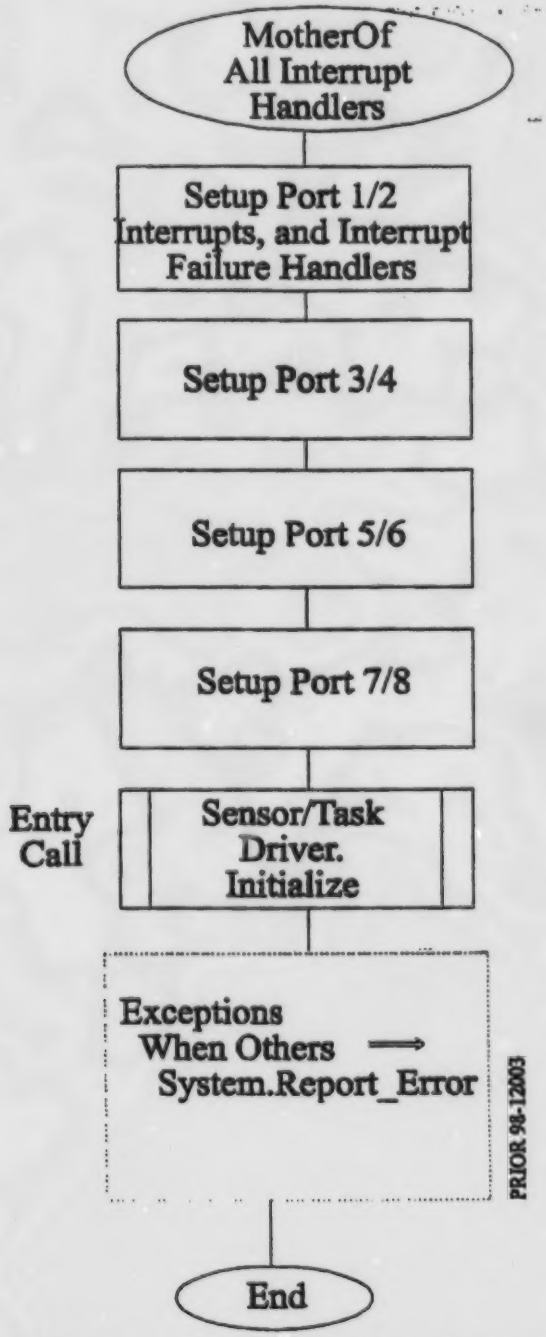


Figure 4.2-3 Mother of All Interrupt Handlers Flow Chart

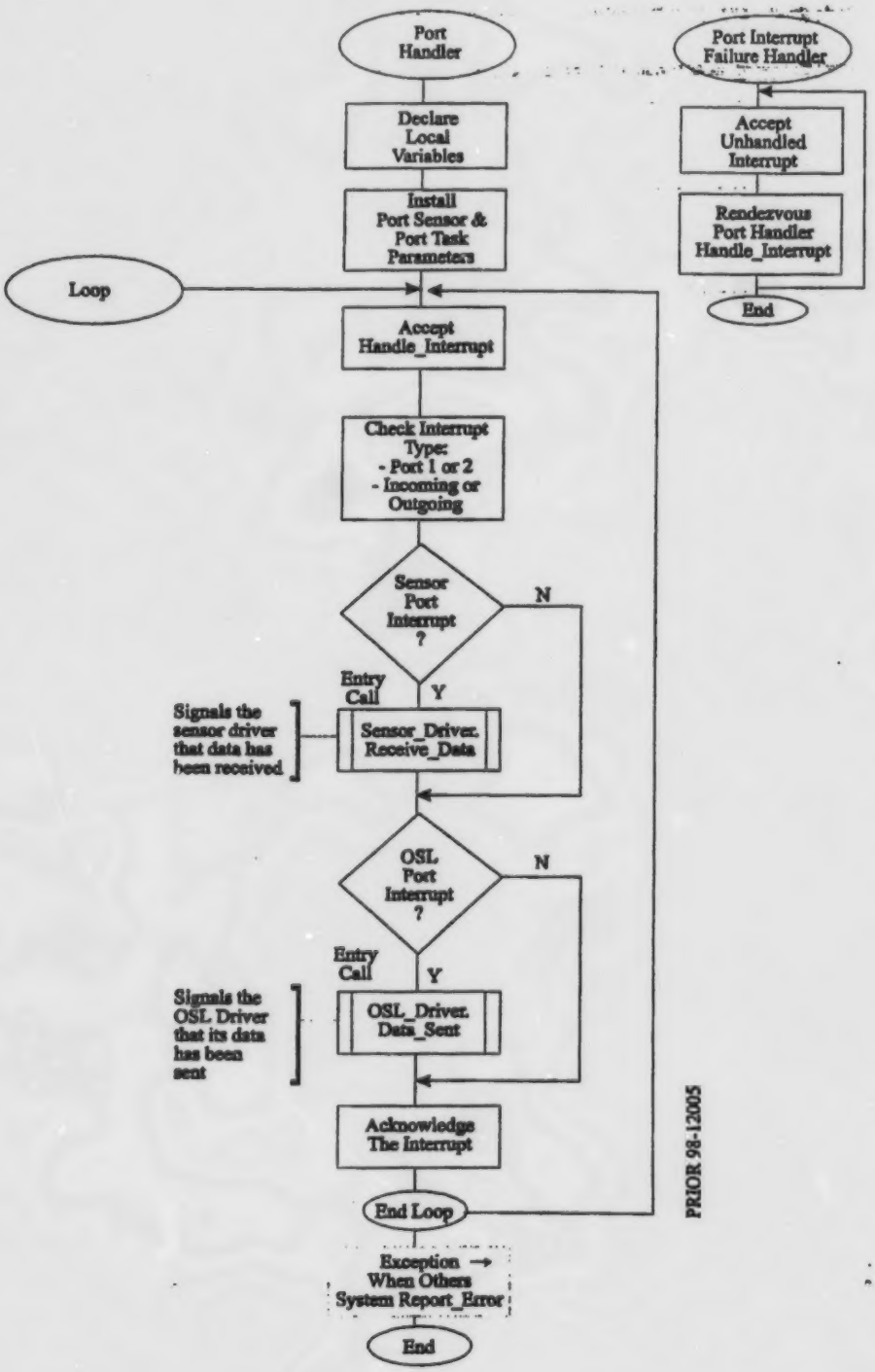


Figure 4.2-4 Port Handlers and Interrupt Failure Handler Flow Chart

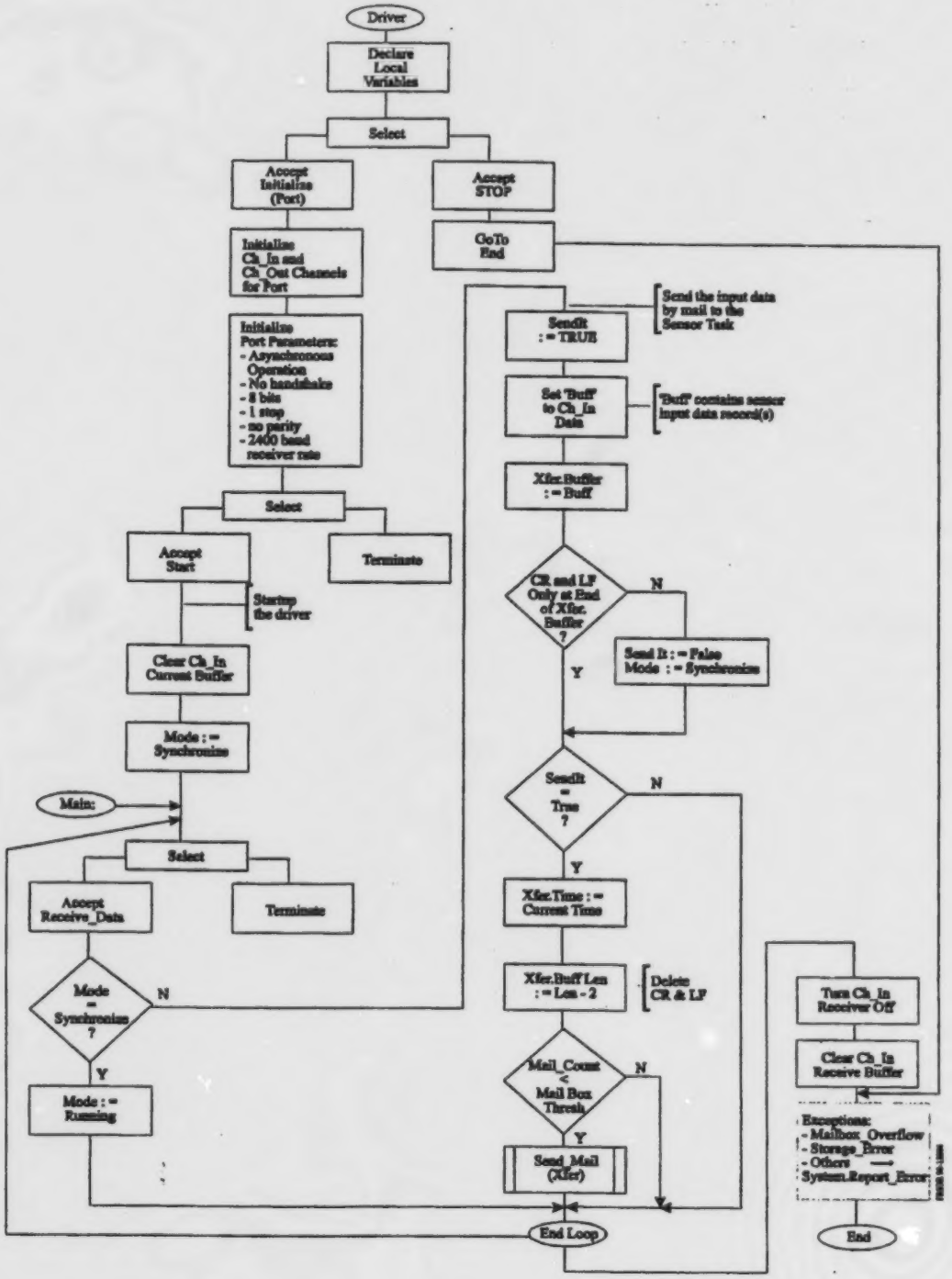


Figure 4.2-5 Device Drivers Flow Chart

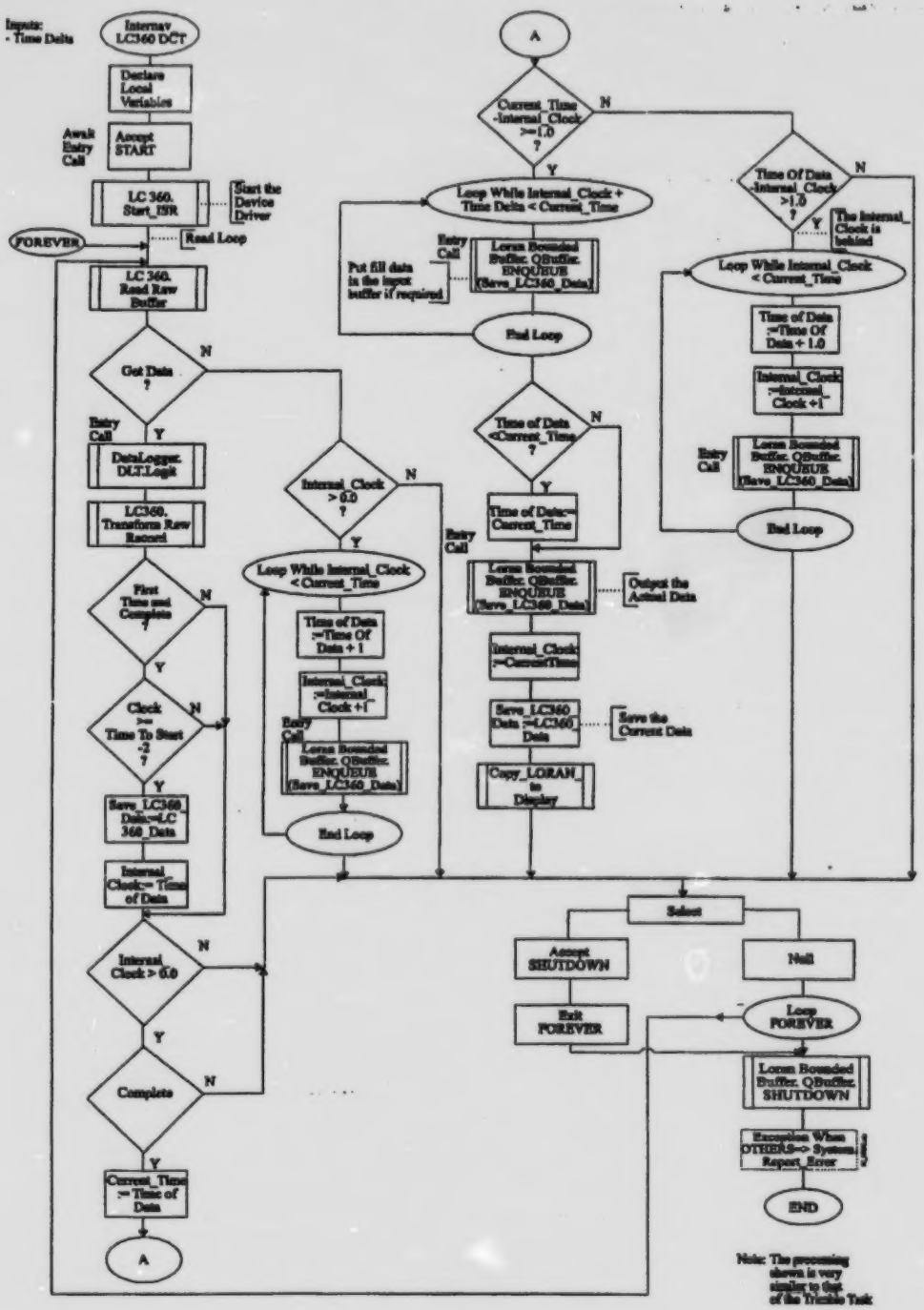


Figure 4.2-6 Real Sensor: Internav LC360 (Loran-C) Flow Chart

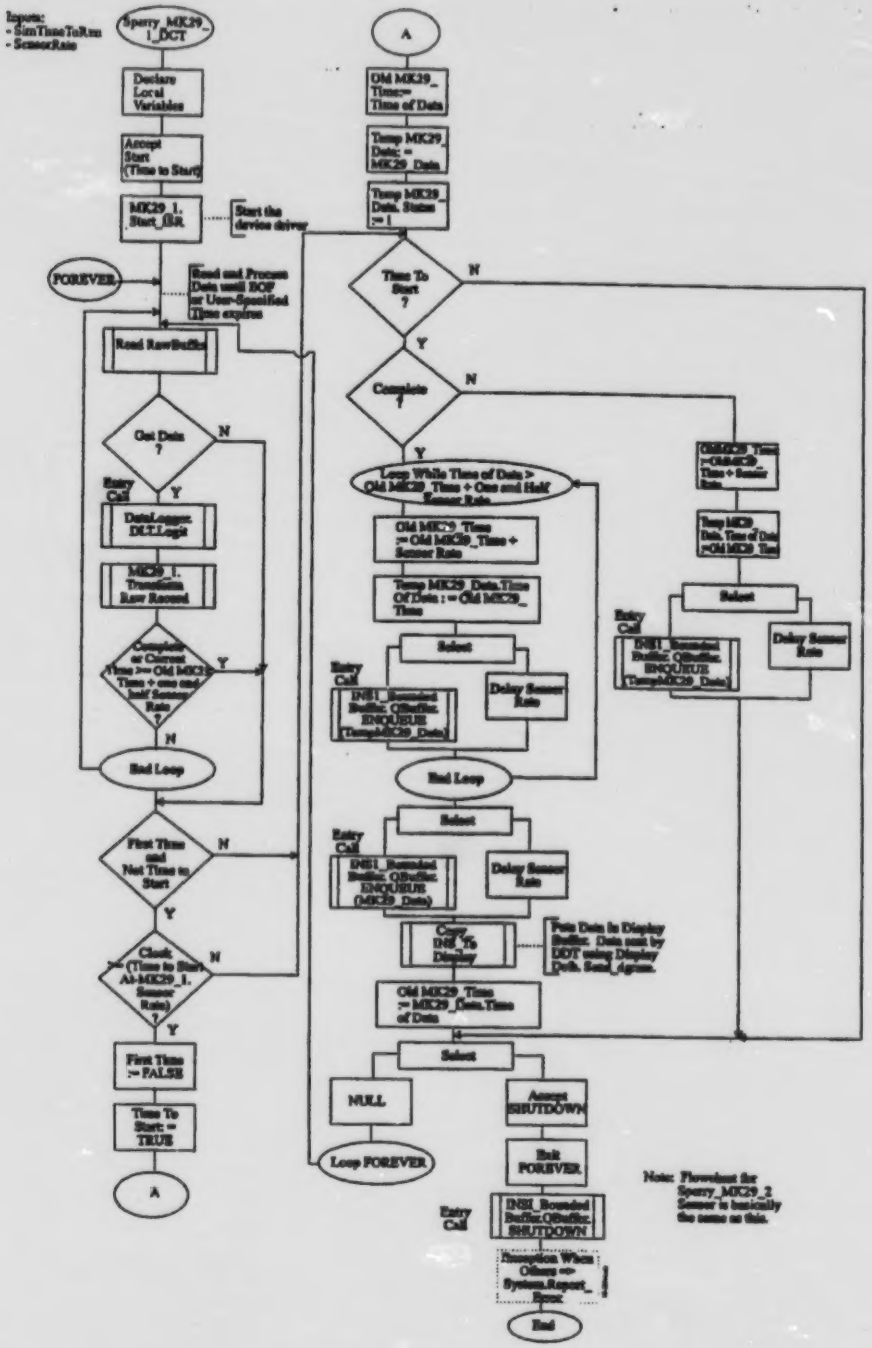


Figure 4.2-7 Real Sensor: Sperry MK29 INS Flow Chart

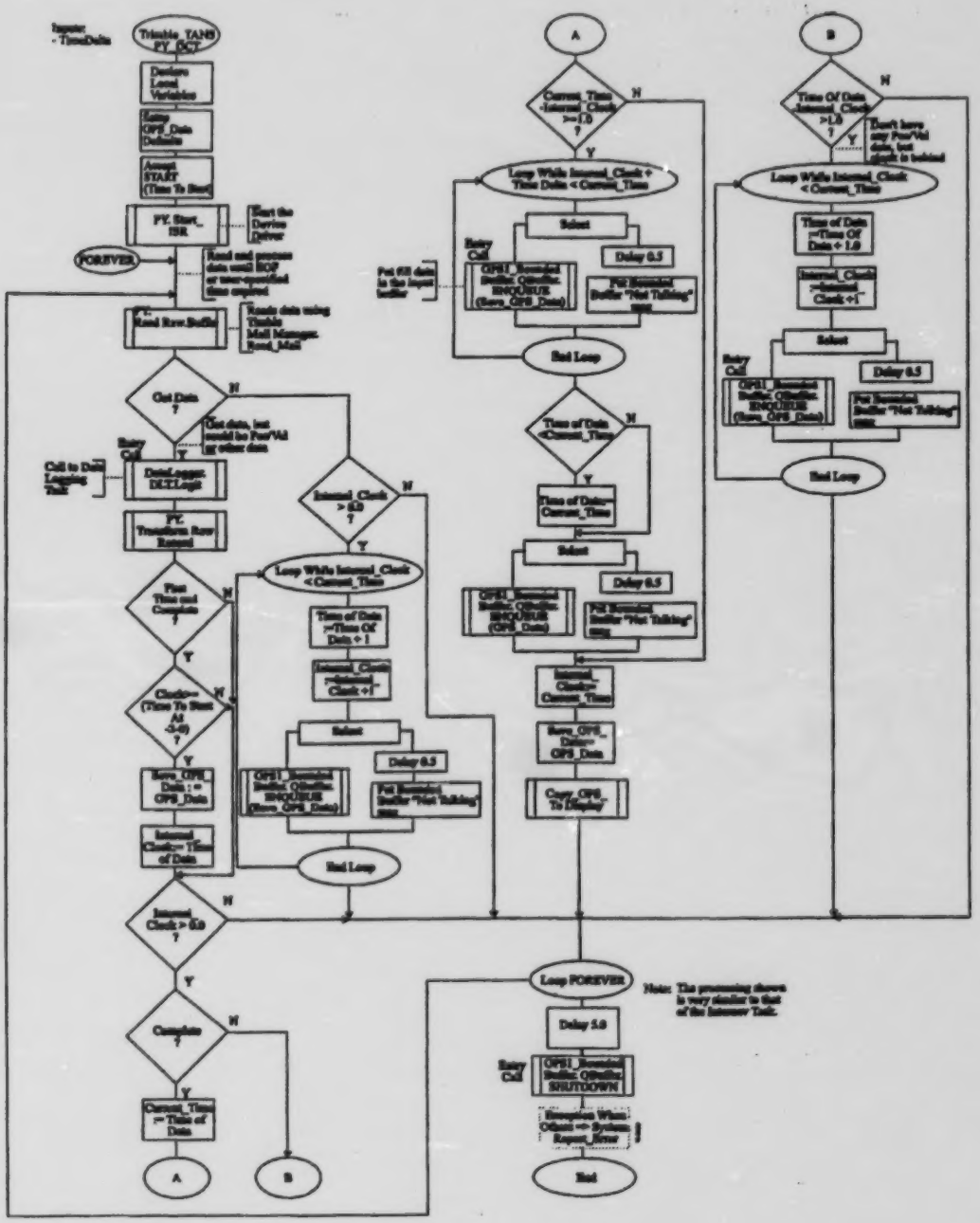


Figure 4.2-8 Real Sensor: Trimble TANS PY (GPS) Flow Chart

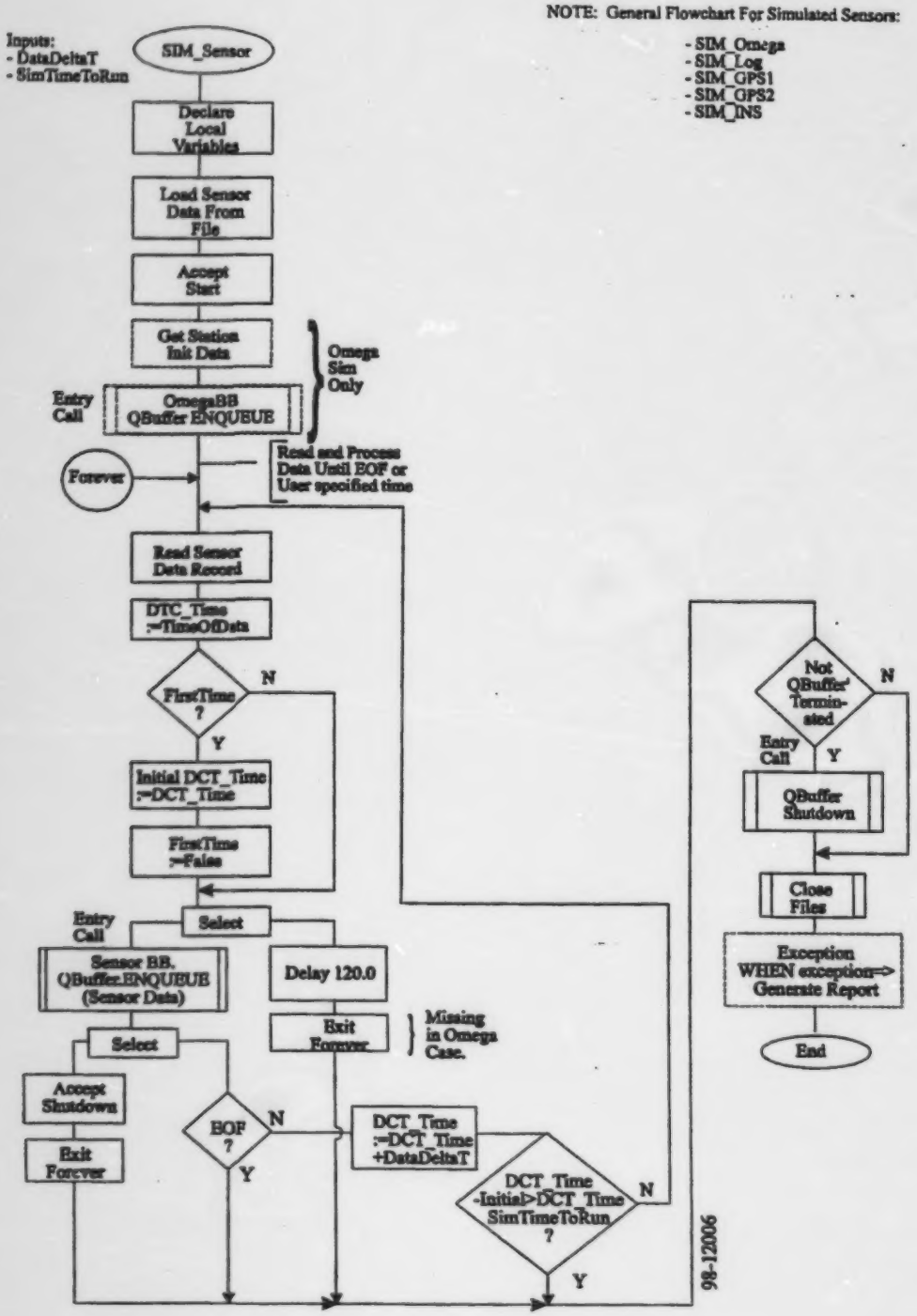


Figure 4.2-9 Simulated Sensors Flow Chart

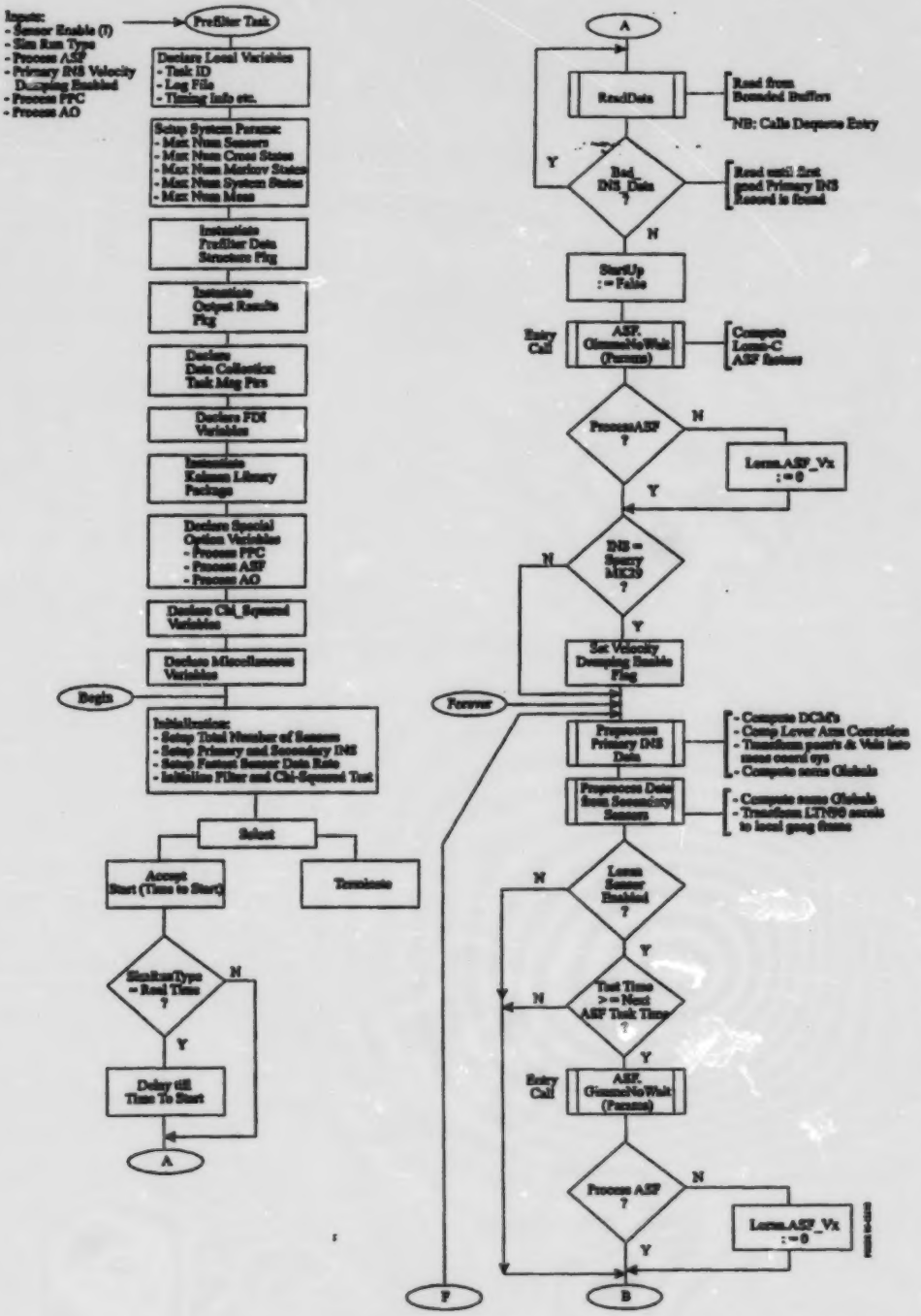


Figure 4.2-10 Prefilter/Filter Flow Chart

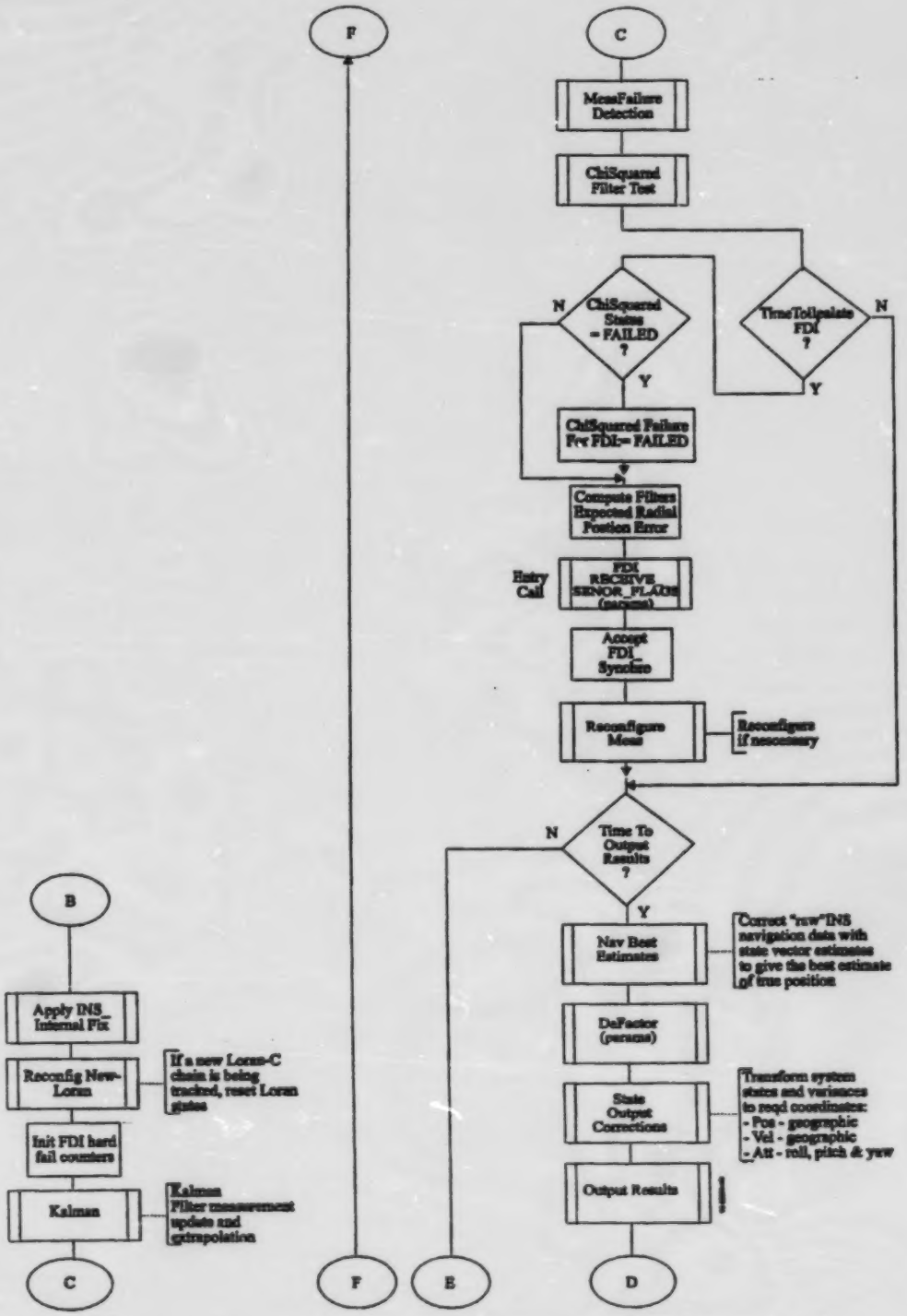


Figure 4.2-11 Prefilter/Filter Flow Chart (Cont'd)

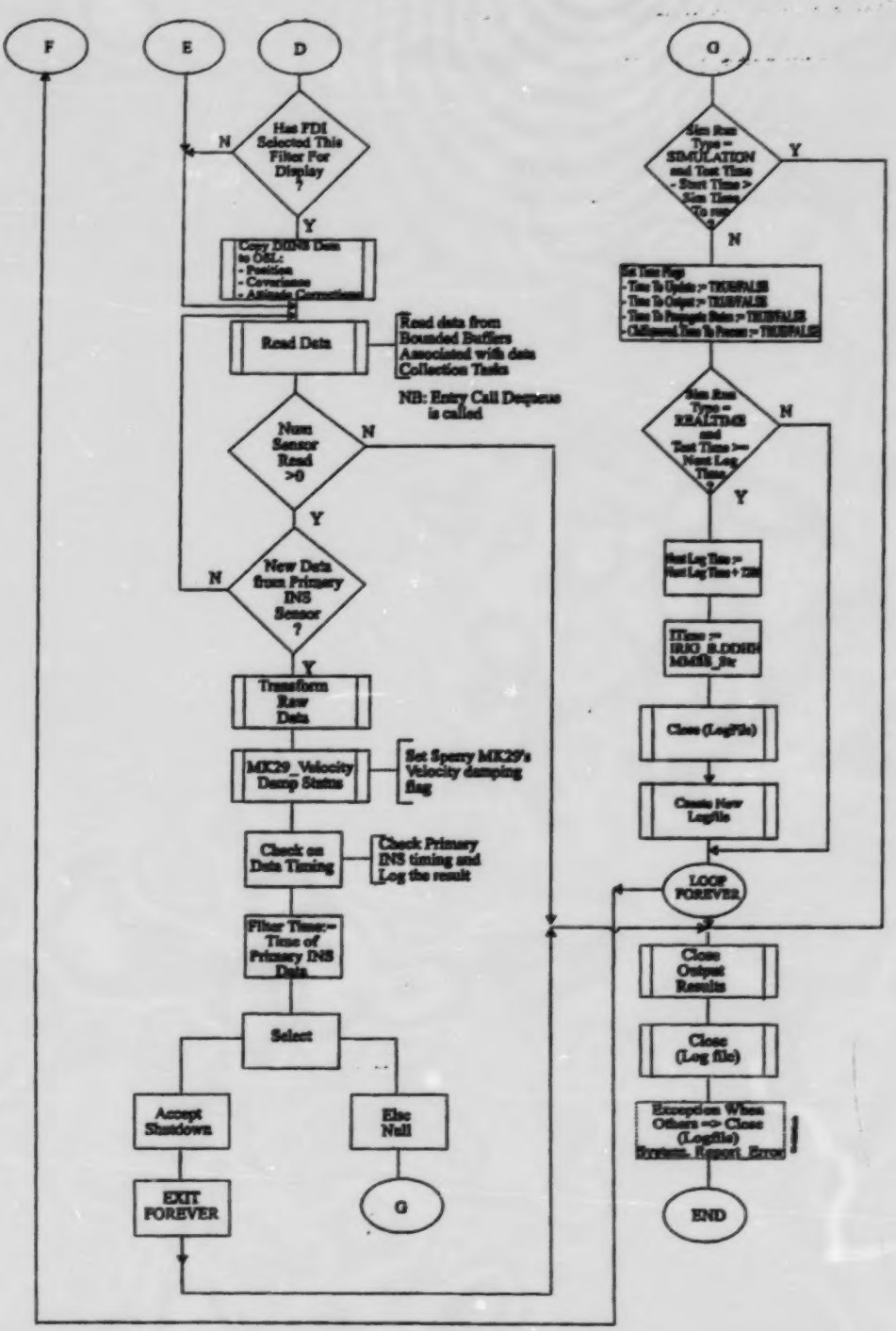


Figure 4.2-12 Prefilter/Filter Flow Chart (Cont'd)

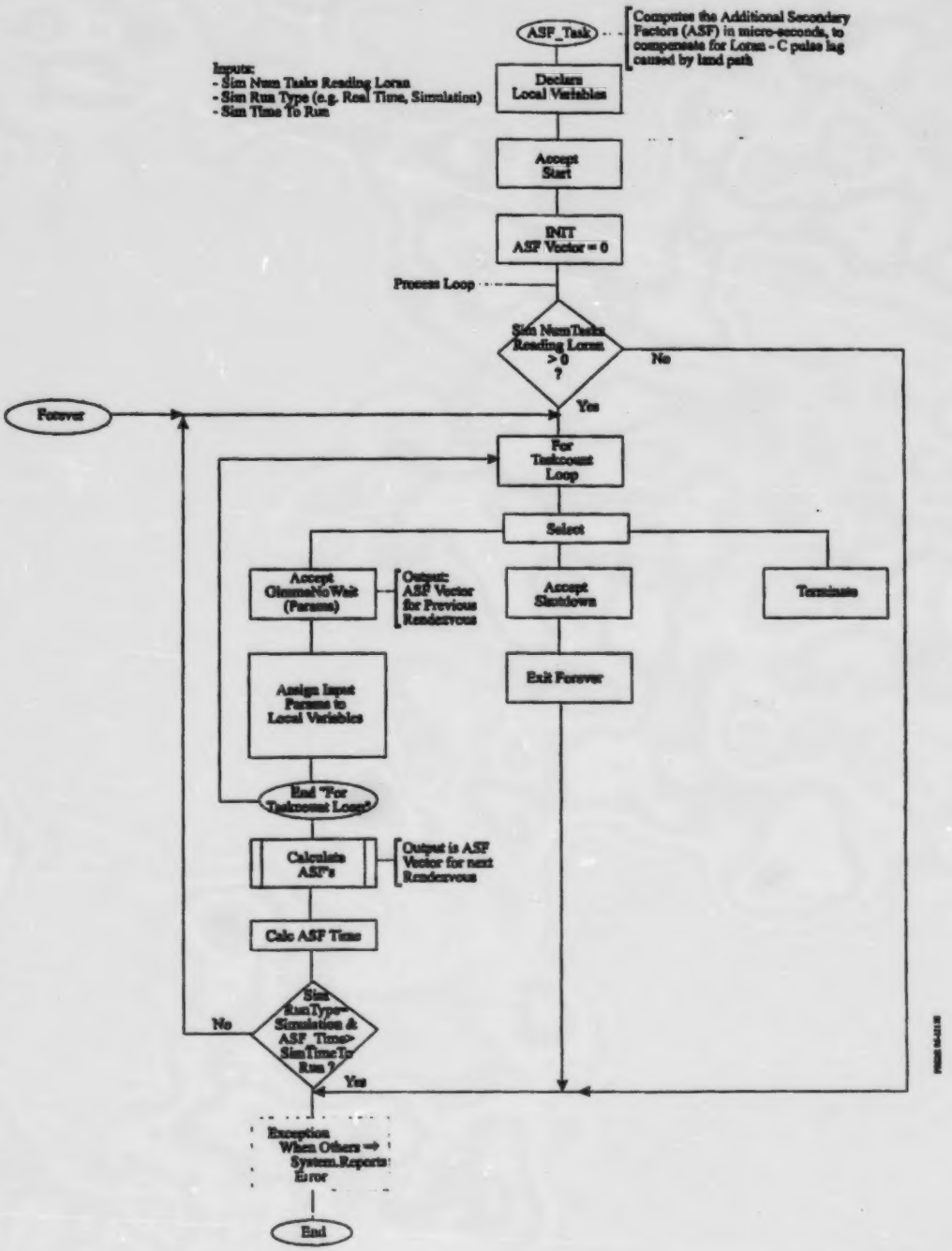


Figure 4.2-13 Additional Secondary Factors Flow Chart

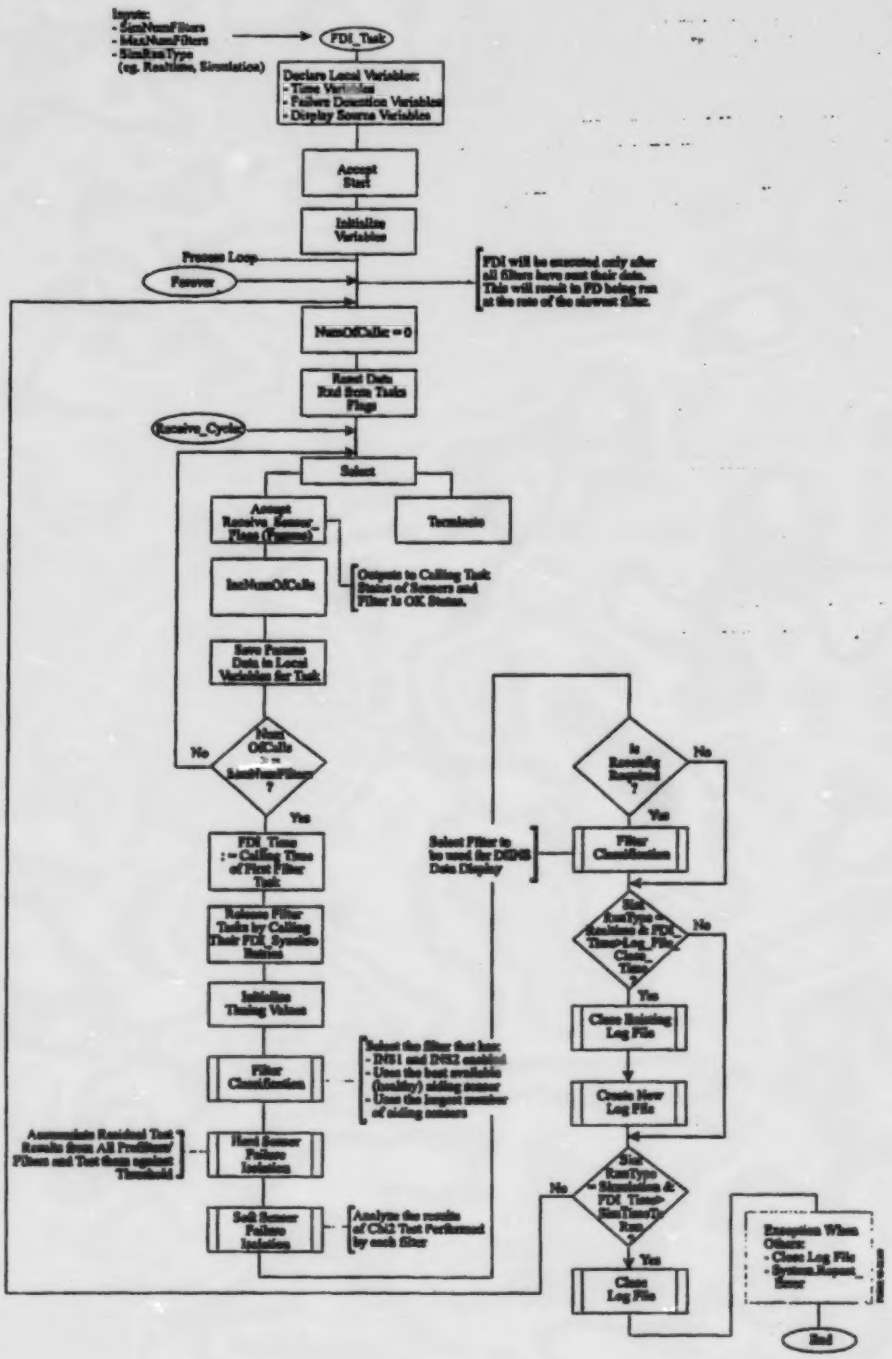

FDI_Task
Declare Local Variables:
Accept Start
Initialize Variables
Process Loop
Forever
FDI will be executed only after all filters have sent their data. This will result in FD being run at the rate of the slowest filter.
NumOfCalls = 0
Reset Data Rcd from Tasks Flags
Receive_Cycle
Select
Accept Receive_Sensor_Data (Params)
Terminate
IncNumOfCalls
Outputs to Calling Task: Status of Sensors and Filter to OK Status.
Save Params Data in Local Variables for Task
Num OfCalls = NumOfFilters ?
No
Yes
FDI_Time := Calling Time of First Filter Task
Release Filter Tasks by Calling Their FDI_Synchro Entries
Initialize Timing Values
Filter Classification
Select the filter that has:
- INS1 and INS2 enabled
- Uses the best available (locality) sliding sensor
- Uses the largest number of sliding sensors
Accumulate Residual Test Results from All Profilers/Filters and Test them against Thresholds
Hard Sensor Failure Isolation
Soft Sensor Failure Isolation
Analyze the results of Chi2 Test Performed by each filter
Is Receiving Enabled ?
No
Yes
Select Filter to be used for DIINS Data Display
Filter Classification
Slot RunType = Simulation & FDI_Time>Log_File_Close_Time ?
No
Yes
Close Existing Log File
Create New Log File
Slot RunType = Simulation & FDI_Time>SimTimeOfRun ?
No
Yes
Close Log File
Exception When Others:
- Close Log File
- System.Report_Error
End

Figure 4.2-14 Failure Detection and Isolation Flow Chart

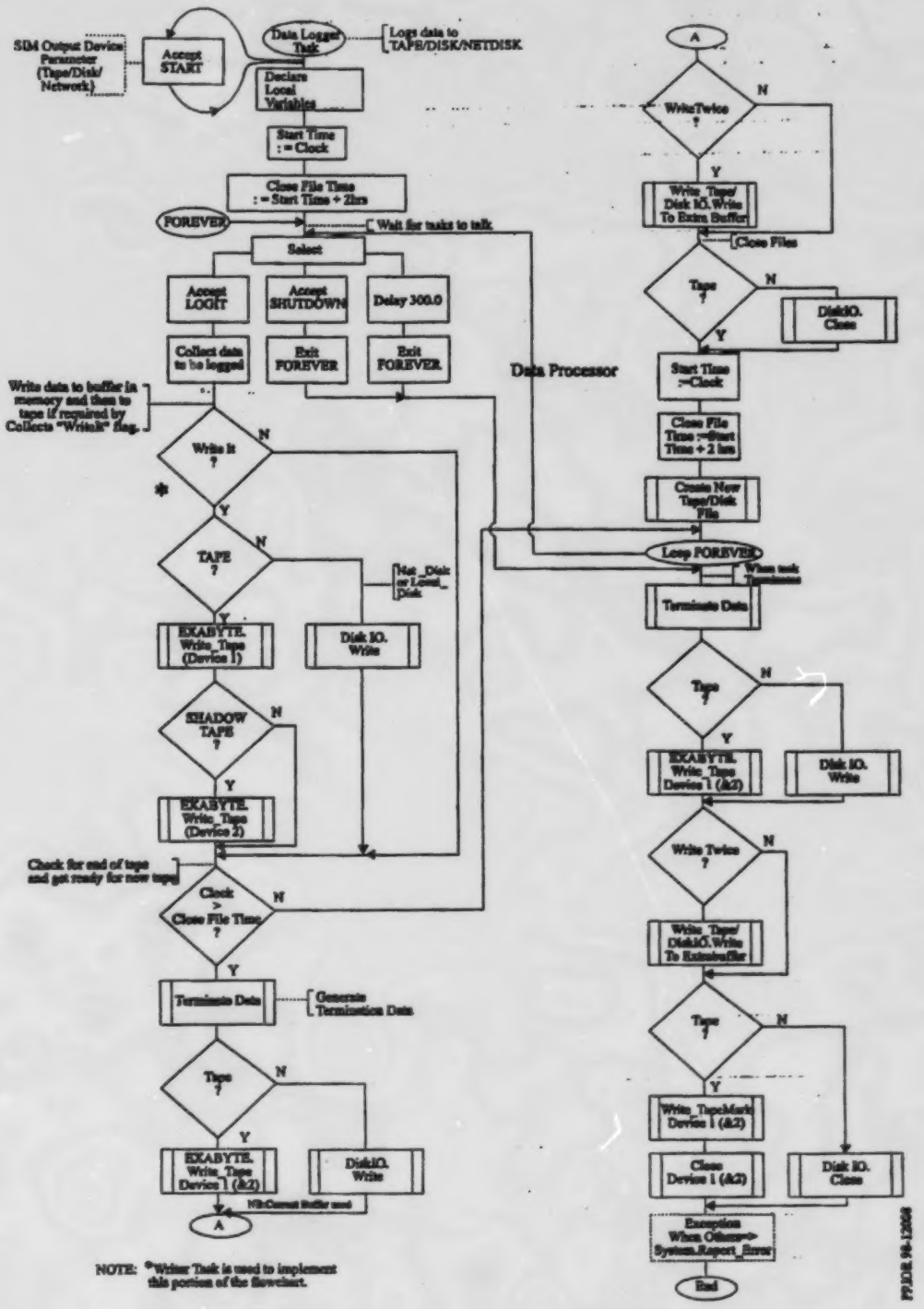


Figure 4.2-15 Data Logger Flow Chart

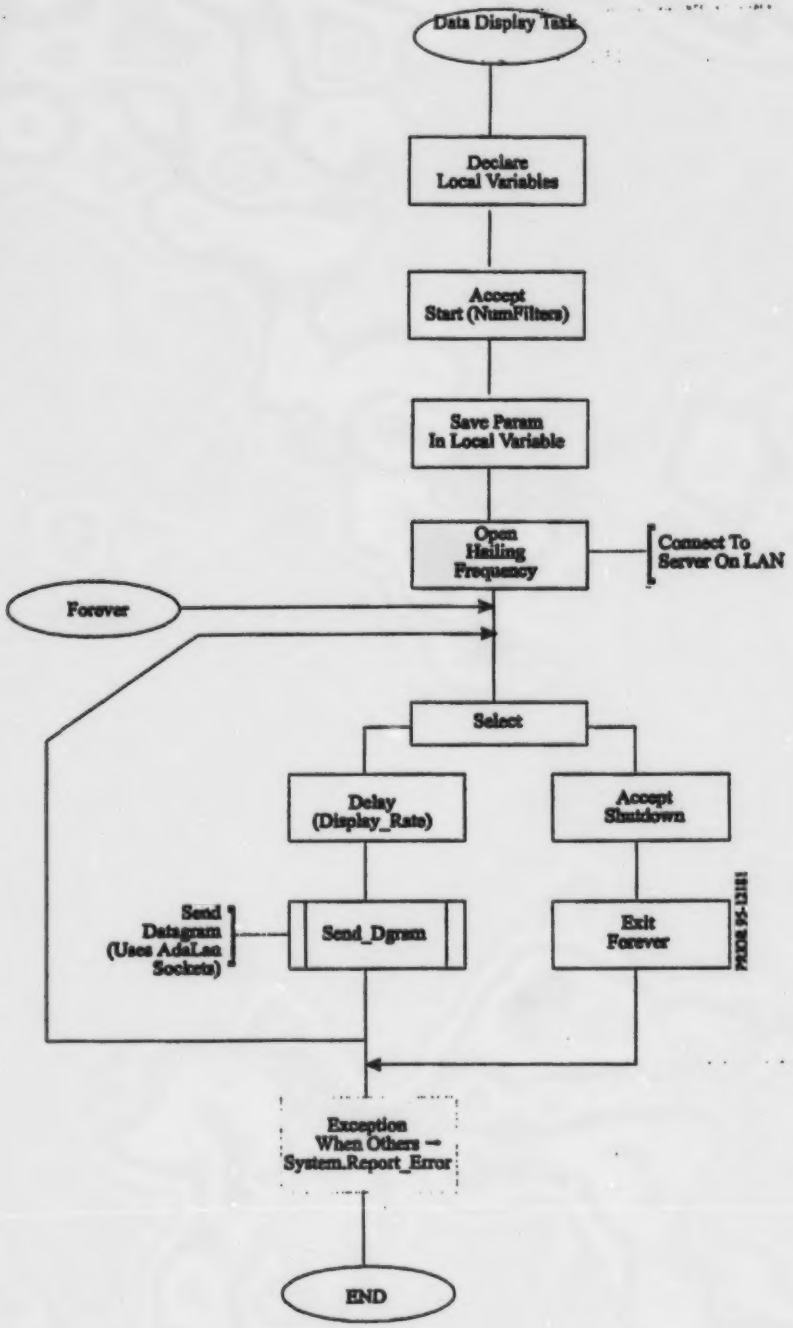


Figure 4.2-16 Data Display Flow Chart

4.3 Interface Design

The DIINS ADM interface design specification is defined in Appendix A to this SDD. The interface interactions between the HCI software and the DIINS Sensor Data Processor software is defined in Appendices B and G respectively.

4.3.1 Interface Identification and Diagrams

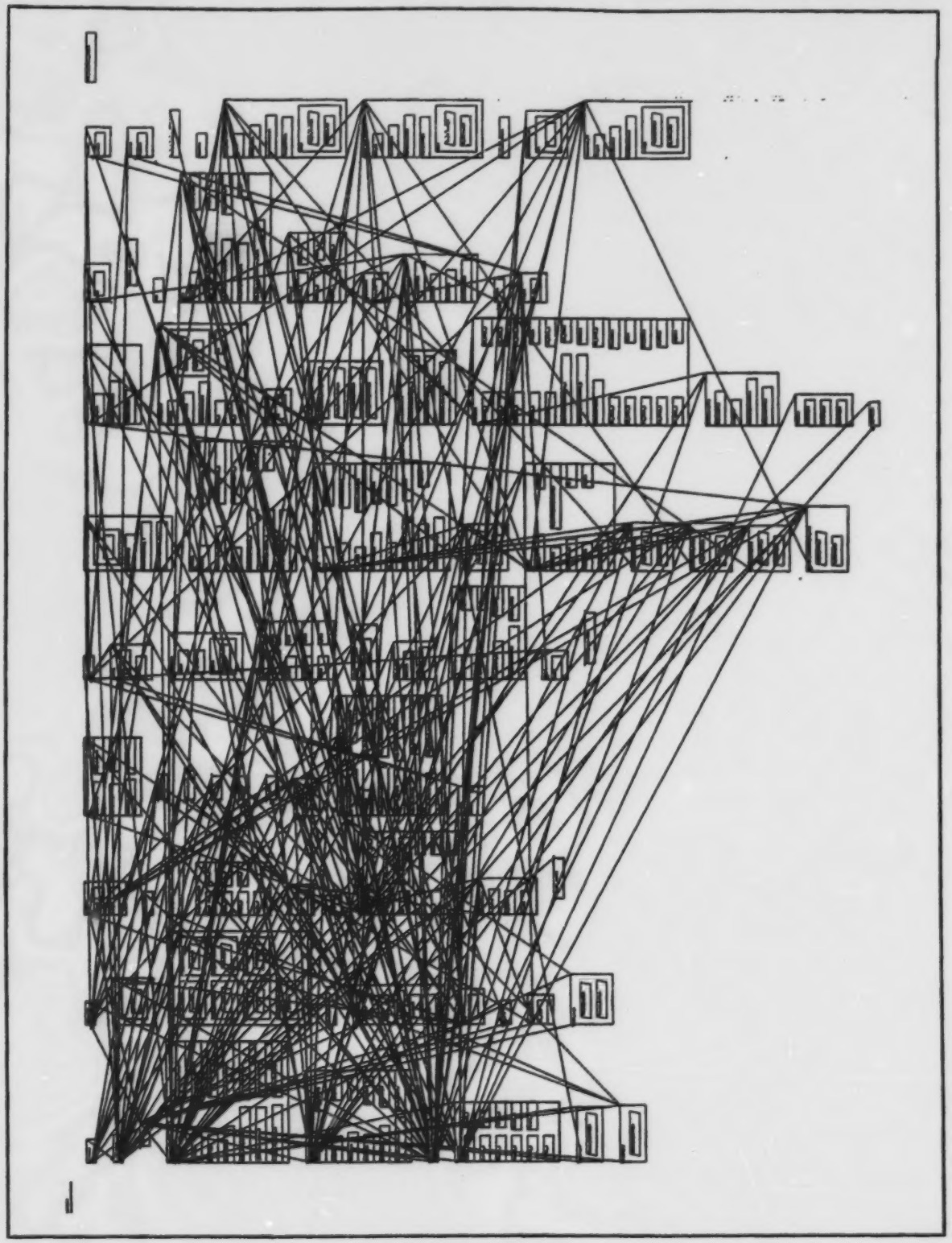
This paragraph has been tailored out.

5. CSCI DETAILED DESIGN

5.1 Overall Software Dependencies

Figure 5.1-1 shows an overall Ada 'Withing' dependency diagram. This diagram was produced with Cadre TeamWork, using the Ada reverse engineering package. Although this diagram shows no detail, it does illustrate the overall structure of the software, the number of packages involved and the number of dependencies between packages.

Section 5.2 to **Section 5.10** present the 'Withing' dependency diagrams and other diagrams as appropriate to define the detailed design for the Ada compilation units to be used in the DIINS ADM software. The software units in the Library, System and Utilities directories have not been put into 'With' diagram form, as in general the units in these directories do not depend on the other units in the directories. The content of these sub directories are listed in Appendix H for completeness. The Data Collection Tasks dependency diagram contains circled numbers that indicate the associated list of 'Withed' packages, (see Appendix I for the details of the packages 'Withed').



5.2 Human Computer Interface, Control and Display CSC

See Appendices A and C for details of the look and behavior of the HCI forms.

The Display Data Task, indicated in Figure 5.2-1, is used to display data from the sensors on the Sun SPARCstation display in the development environment only. See Figure 4.2-16 for the detailed design of the data display task shown in Figure 5.2-1.

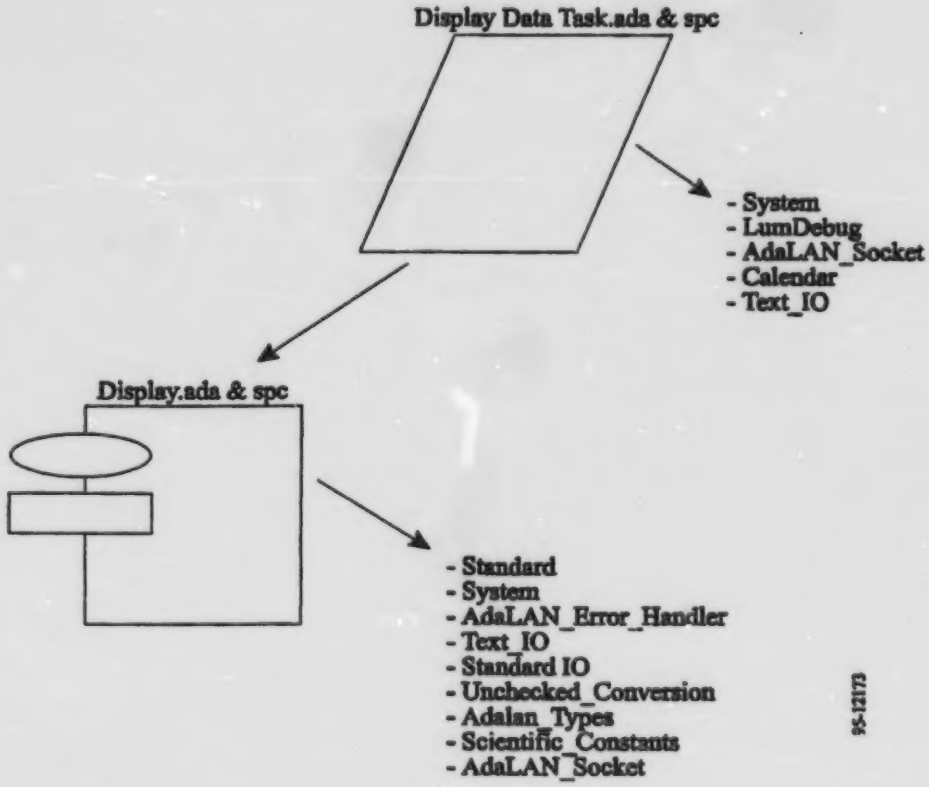


Figure 5.2-1 Display Data Task (DDT) Ada Unit Dependency Diagram

5.3 Task Master CSC

See **Figure 5.7-1** to understand the role of the Task Master CSC. The detailed design of the TaskMaster CSC is provided in **Figure 4.2-2.**

5.4 Device Drivers CSC

See **Figure 4.2-5** and Section 5.7 for details of the Device Driver CSC design.

5.5 Encoders CSC

See Section 5.6 for the details of the design of the Encoder CSC.

5.6 Data Collector CSC

See **Figure 5.6-1** and **Figure 5.6-2** for details of the Data Collector CSC design.

Figure 5.6-1 illustrates the detailed design for an enhanced output interface. This design is a "cleaned-up" or simplified version of the current design. The two basic functions of the output interface have been captured in the Data Collector and NMEA Encoder packages. By isolating these functions in the design the resulting code will be easier to adapt to meet new or changing requirements. The Collect_Data procedure in the Data Collector package receives sensor data or error messages in the form of a variant record containing the individual sensor data elements or error message string. The data record is placed in a queue. The Queue is an instantiation of the generic queue package. The Collector_Master task will wait for data on the queue. When data is available, the Collector_Master will retrieve the data record from the queue and pass it to the Encode procedure in the NMEA Encoder package (and to other encoder packages as required) to be formatted into a data sentence. The routines to format the individual sensor data elements will be encapsulated within the procedure Encode. The main portion of the Encode procedure will use these routines to construct the data sentences to be sent to the OSL Driver. The Mailbox mechanism which is used to pass data between the NMEA and OSL Driver in the current implementation may be replaced by a queue mechanism encapsulated in the OSL Driver, as indicated in **Figure 5.6-1.**

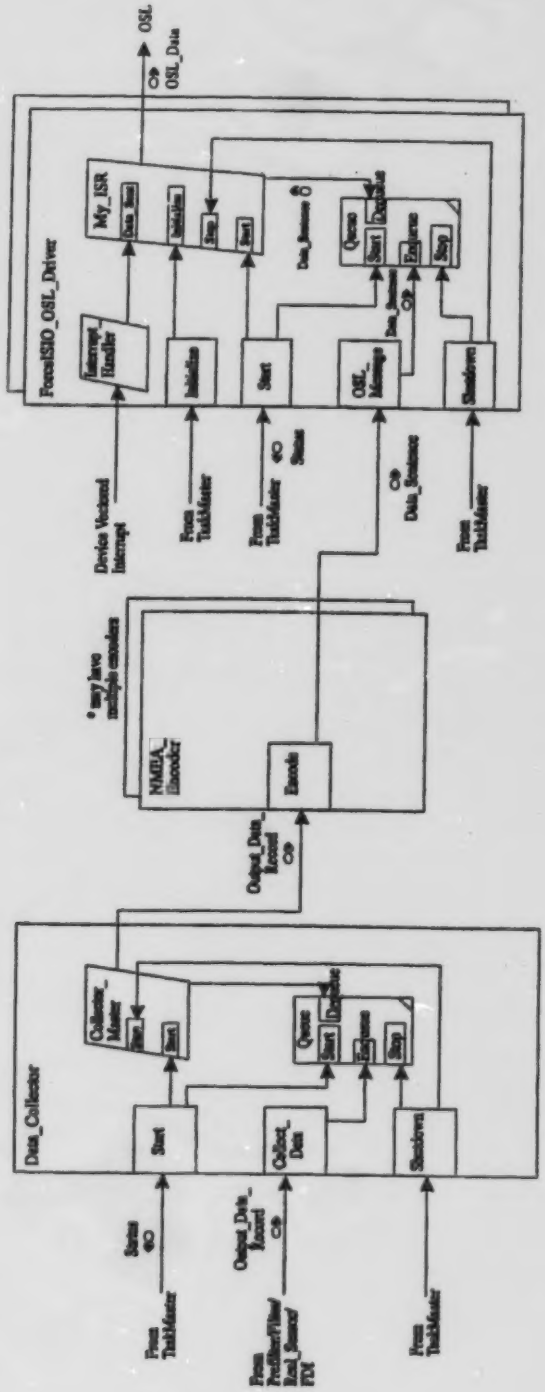


Figure 5.6-1 Data Collector CSC Design

The specifications for the Data Collector and NMEA Encoder packages would resemble the following:

```
with Global_Definitions;

package Data_Collection_Records is

   type Output_Data_Type is
      (Void_Data,
       Sensor_Data,
       Filter_Data,
       Error_Data);

   type Output_Record_Type
      (Output_Data : Output_Data_Type := Void_Data) is
      record
         case Output_Data is

            when Void_Data =>
               null;

            when Sensor_Data =>
               Sensor :
                  Global_Definitions.Sensor_Data_Type;

            when Filter_Data =>
               Filter :
                  Global_Definitions.Filter_Data_Type;

            when Error_Data =>
               Error_Message :
                  Global_Definitions.Error_Message_Type;
         end case;

      end record;

end Data_Collection_Records;


with Data_Collection_Records;

package Data_Collector is

   ------------------------------------------------
   -- Initialize data collector package.
```

```
   ------------------------------------------------
   procedure Start (Status : out Boolean);
   ------------------------------------------------
   -- Shutdown data collector package.
   ------------------------------------------------

   procedure Shutdown;

   ------------------------------------------------
   -- Receive output data.
   ------------------------------------------------

   procedure Collect_Data
      (Output_Data_Record : in Data_Collection_Records.
          Output_Record_Type);

end Data_Collector;


with Data_Collection_Records;

package NMEA_Encoder is

   ------------------------------------------------
   -- Encode output data.
   ------------------------------------------------

   procedure Encode
      (Output_Data_Record : in Data_Collection_Records.
          Output_Record_Type);

end NMEA_Encoder;
```

Figure 5.6-2 illustrates the dependencies between the software modules used in the implementation of the data logging functionality in the DIINS ADM. These modules provide for the logging of information to disk or tape, provided that the relevant storage device is available to the DIINS ADM.

Figure 4.2-15 illustrates the detailed design of the Data Logger task used in **Figure 5.6-2.**

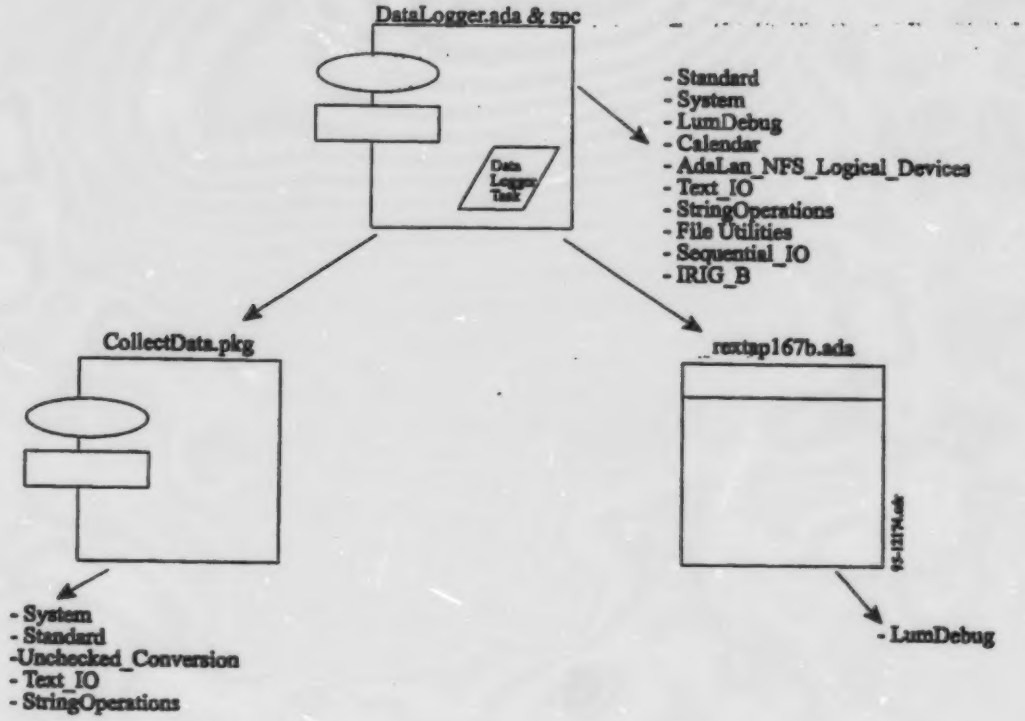


Figure 5.6-2 Data Logger Task (DLT) Ada Unit Dependency Diagram

5.7 Sensor CSC

Figure 5.7-1 illustrates the sensor data collection tasking architecture used in the DIINS XDM. This same basic tasking architecture will be used on the DIINS ADM. The main difference for the ADM will be that the sensor tasks will be derived from a generic sensor task type, as described in Figure 5.7-2 and its accompanying text, and not using separate task implementations for each sensor, as was the case in the XDM.

See **Figures 4.2-6** to **4.2-9** for the detailed design of the Sensor CSC tasks.

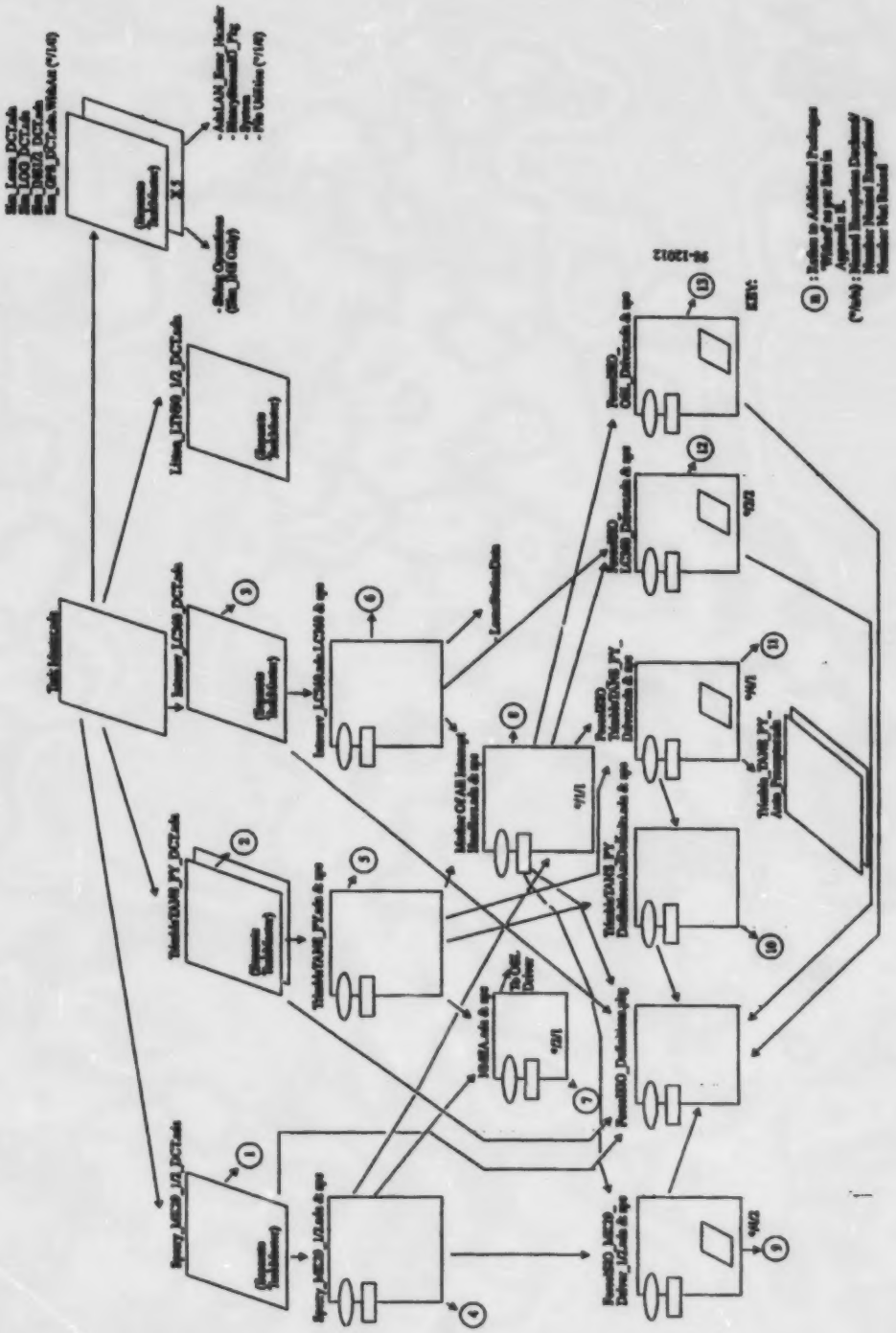


Figure 5.7-1 Sensor Data Collection Tasks (DCT) Ada Unit Dependency Diagram

Figure 5.7-2 illustrates the detailed design for a generic Real Sensor package. In this design the Read_Raw_Buffer procedure receives raw sensor data from the associated device driver. This data is placed in a queue. The Queue is an instantiation of the generic queue package. The Sensor_Master task will wait for data on the queue. When data is available, the Sensor_Master will retrieve the raw sensor data from the queue. The Transform_Raw_Record procedure inside the Sensor_Master task will perform calculations on the raw sensor data and transform the data into a sensor data record. The sensor data record will then be passed to the appropriate Bounded Buffer package and to the Data Collector package. Any error messages generated from the Real Sensor package will be passed to the FDI package. The package's tasks will be hidden within the body of the package. Encapsulation of tasks reduces the amount of extraneous information in the package specification and conceals task entries that should not be made public. The Start and Shutdown procedures will control access to the Start and Stop entries in the tasks. The Start procedure will rendezvous with the Start entries in the tasks. If a task fails to start (i.e., an exception is raised), a false status will be returned to the Task Master. The Shutdown procedure will rendezvous with the Stop entries in the tasks.

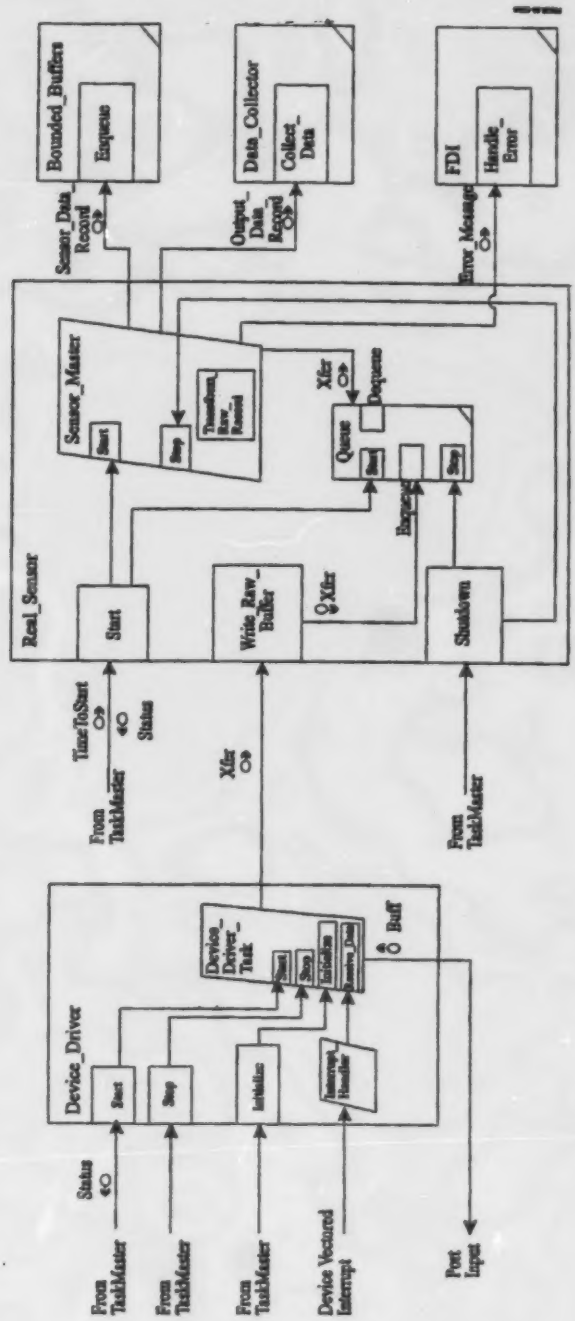


Figure 5.7-2 Sensor CSC Design

The specification for this package would resemble the following generic specification:

```
generic

   -- Data type for raw sensor data from device driver.

   type Raw_Sensor_Data_Type is private;

   -- String identifier for sensor.

   Sensor_Identifier : in String;

   -- Procedure to transform raw sensor data.

   with procedure Transform_Raw_Record
      (Raw_Sensor_Data : in Raw_Sensor_Data_Type);

package Real_Sensor is

   ------------------------------------------------
   -- Initialize sensor package.
   ------------------------------------------------

   procedure Start (TimeToStart : in     Duration;
                    Status      :    out Boolean);

   ------------------------------------------------
   -- Read raw sensor data from device driver.
   ------------------------------------------------

   procedure Read_Raw_Buffer
      (Raw_Sensor_Data : in Raw_Sensor_Data_Type);

   ------------------------------------------------
   -- Shutdown sensor package.
   ------------------------------------------------

   procedure Shutdown;

end Real_Sensor;
```

To instantiate this package, a data type for the raw sensor data and a procedure to transform the raw sensor data must first be defined. The data type and procedure would then become input parameters in the instantiation of the generic. A string identifier for the sensor is also passed in as

a parameter to be used in the generation of any error messages. As an example, an instantiation of the Real Sensor package for an INS sensor would resemble the following:

```
type XferREC is
   record
      Time    : Duration;
      BuffLen : Integer;
      Buffer  : BuffArray;
   end record;

procedure Transform_Raw_INS is

   begin
      .
      .
      .
   end Transform_Raw_INS;

package Real_INS is new Real_Sensor
   (XferREC, "INS1", Transform_Raw_INS);
```

5.8 Bounded Buffer CSC

See Figures 4-1 and 5.7-1 for details of the implementation of the Bounded Buffers CSC.

5.9 Prefilter/Filter CSC

5.9.1 Low Rate Data Prefilter/Filter

Figure 5.9-1 and **Figure 5.9-2** illustrate the Ada unit dependencies associated with the Prefilter/Filter CSC.

See **Figures 4.2-10** to **4.2-13** for the detailed design of the Prefilter/Filter CSC components.

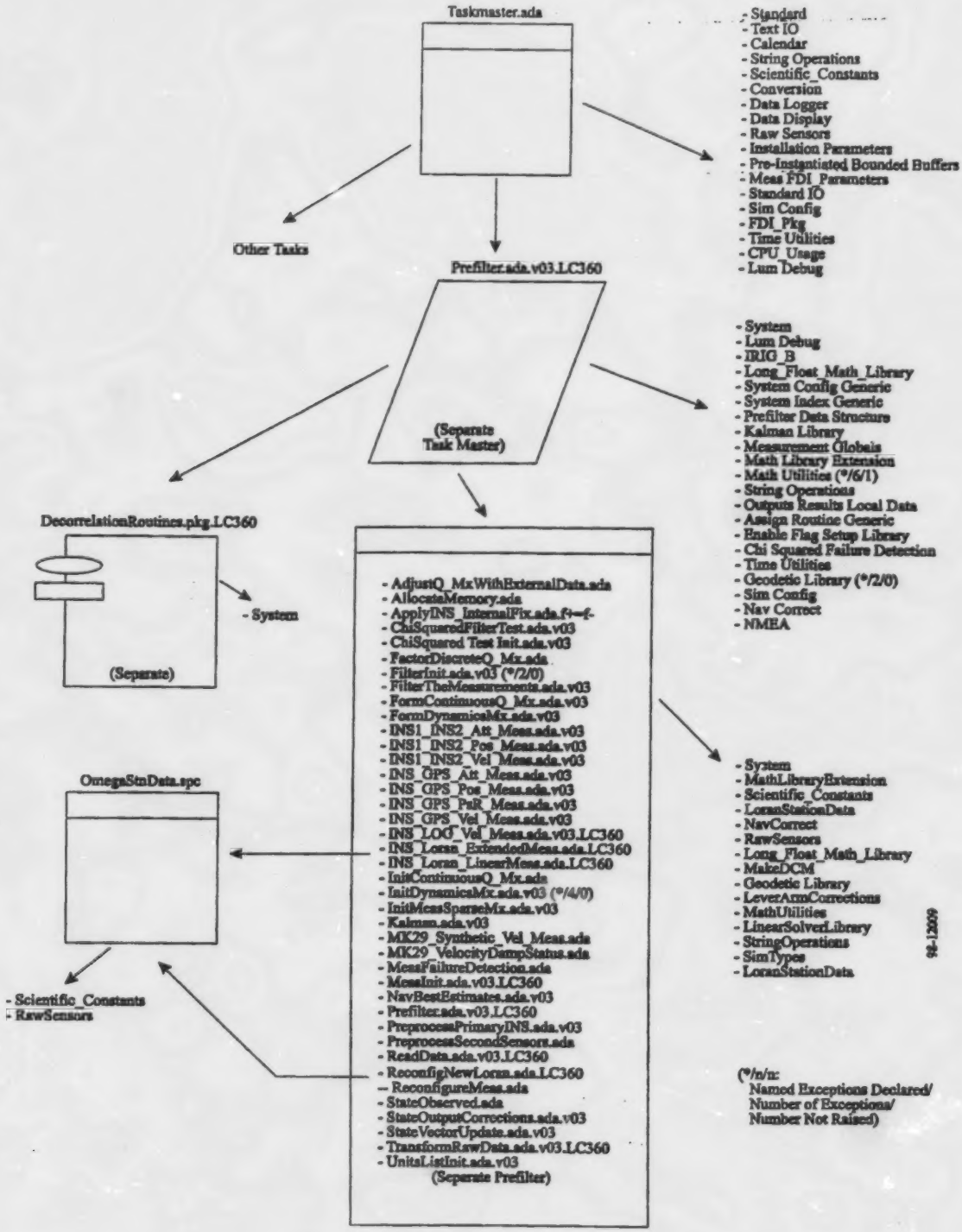


Figure 5.9-1 Unifilter Ada Unit Dependency Diagram

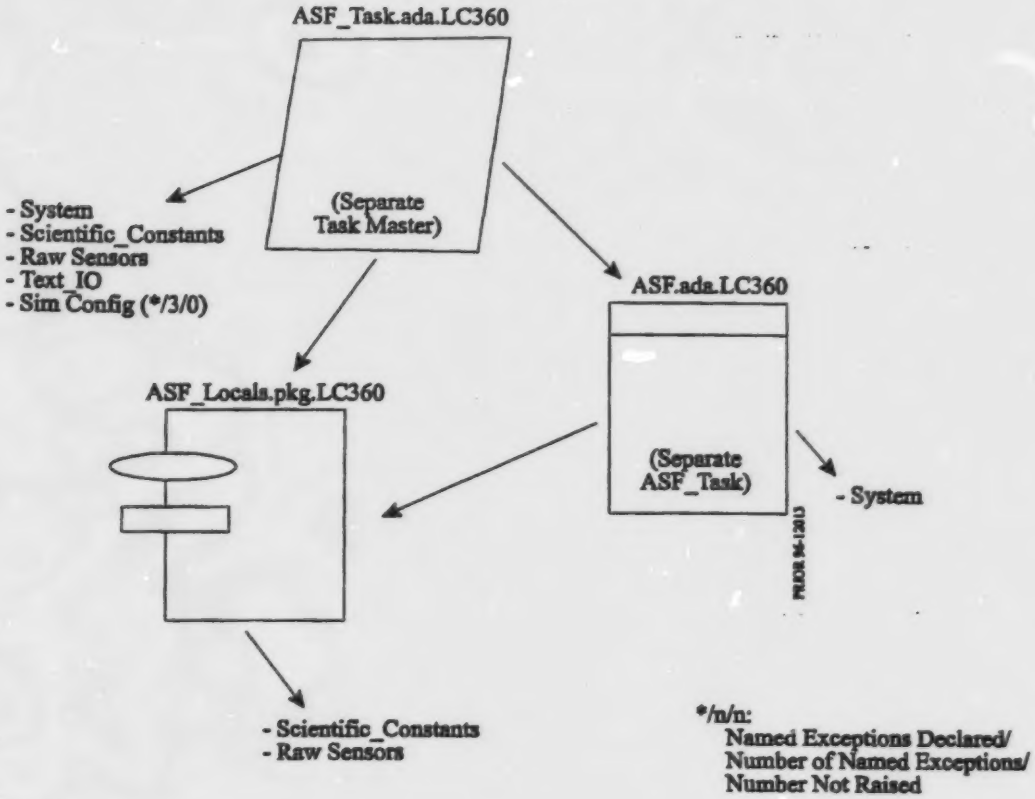


Figure 5.9-2 Additional Secondary Factors (ASF) Ada Unit Dependency Diagram

5.9.2 High Rate Attitude Data Filter

Figure 5.9-3 illustrates the processing performed in order to distribute the high rate attitude data.

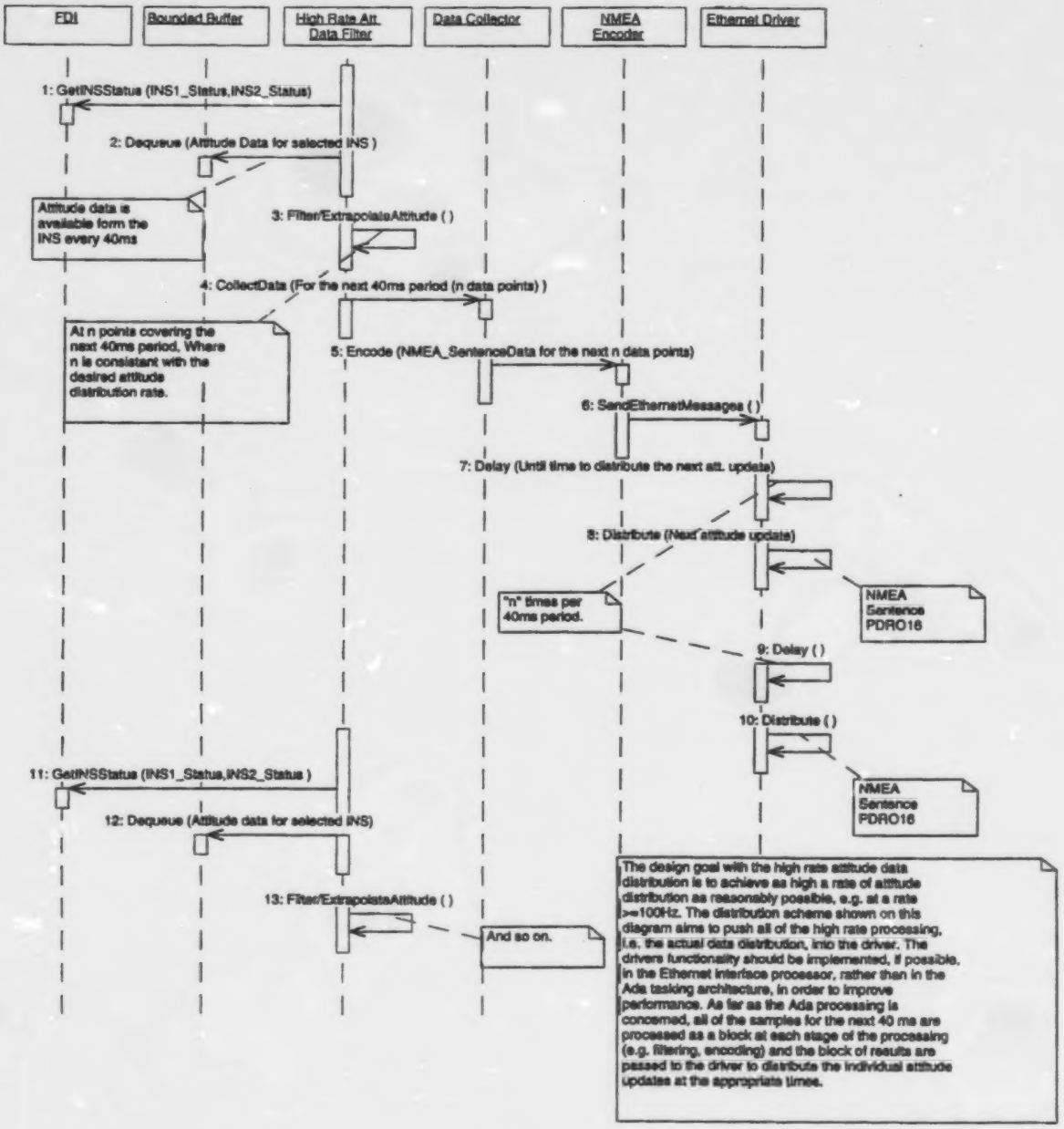

FDI
Bounded Buffer
High Rate Att. Data Filter
Data Collector
NMEA Encoder
Ethernet Driver
1: GetINSStatus (INS1_Status,INS2_Status)
2: Dequeue (Attitude Data for selected INS)
Attitude data is available form the INS every 40ms
3: Filter/ExtrapolateAttitude ( )
4: CollectData (For the next 40ms period (n data points))
At n points covering the next 40ms period, Where n is consistant with the desired attitude distribution rate.
5: Encode (NMEA_SentenceData for the next n data points)
6: SendEthernetMessages ( )
7: Delay (Until time to distribute the next att. update)
8: Distribute (Next attitude update)
"n" times per 40ms period.
NMEA Sentence PDRO16
9: Delay ( )
10: Distribute ( )
NMEA Sentence PDRO16
11: GetINSStatus (INS1_Status,INS2_Status)
12: Dequeue (Attitude data for selected INS)
13: Filter/ExtrapolateAttitude ( )
And so on.
The design goal with the high rate attitude data distribution is to achieve as high a rate of attitude distribution as reasonably possible, e.g. at a rate >=100Hz. The distribution scheme shown on this diagram aims to push all of the high rate processing, i.e. the actual data distribution, into the driver. The drivers functionality should be implemented, if possible, in the Ethernet interface processor, rather than in the Ada tasking architecture, in order to improve performance. As far as the Ada processing is concerned, all of the samples for the next 40 ms are processed as a block at each stage of the processing (e.g. filtering, encoding) and the block of results are passed to the driver to distribute the individual attitude updates at the appropriate times.

Figure 5.9-3 High Rate Attitude Data Distribution Sequence Diagram

5.10 Failure Detection and Isolation Controller CSC

Figure 5.10-1 illustrates the Ada unit dependencies for the FDI CSC.

Figure 4.2-14 illustrates the detailed design of the FDI task.

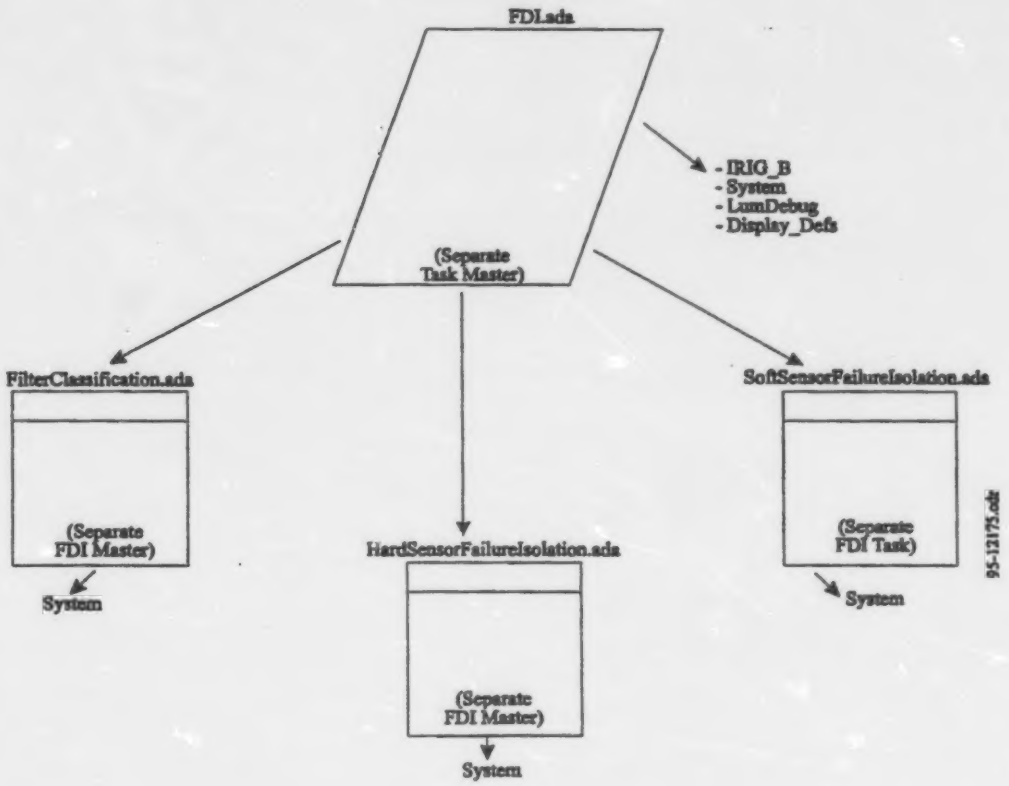


Figure 5.10-1 Failure Detection and Isolation (FDI) Ada Unit Diagram

5.11 System Configuration Controller CSC

See Appendix G for the detailed design of the System Configuration Controller CSC.

6. REQUIREMENTS TRACEABILITY

6.1 Introduction

Sections 4.1.3 to 4.1.11 provide the traceability for each DIINS ADM CSC to the requirements in the System Specification(Reference [2-3]) that have been allocated to them, hence, which they satisfy.

This section provides the traceability of the system software requirements (from Reference [2-3]) to the CSCs, the reverse traceability, for ease of reference.

The section numbering used in the following sections corresponds to those used in Appendix A of the DIINS ADM Development Phase 1 Report (Ref [2-4]). The parent requirement trace is to the requirements as specified in the Request For Proposal (Ref[2-1]).

SYSTEM FUNCTIONAL REQUIREMENTS TRACEABILITY

1. System Configuration

A	The DIINS system shall be a subsystem of SHINNADS.	A.2	Hardware (H)
B	DIINS will reside within the SHINNADS console.	A.2	H
C	Communication between the DIINS processor and the SHINNADS processor will be via a link internal to the console.	A.2	H & S 1.1.2 a Device Drivers CSC
D	In the ADM, this link shall be a two way RS-422 link to allow for DGPS data to be provided to DIINS from SHINNADS, and in addition, to allow for complete control and display of the DIINS internal parameters and sensor and filter configuration.	A.2	H & S 1.1.2 c Device Drivers CSC

2. System Components.

A	*Processing Component.* This unit shall consist of Computer Card Assemblies (CCA's) compatible with the SHINNADS card cage.	A.3.1	H
B	The components chosen shall have sufficient processing capacity to run all the DIINS software under a maximal sensor configuration and at least TBD(15) parallel Kalman filters.	A.3.1	H
C	*Sensor Interface Component.* This unit shall consist of high speed CCA's capable of reading both RS232 and RS422 serial data from the various sensors.	A.3.2	H & S 1.1.2 c Device Drivers CSC
D	*Control and Display Component.* This shall consist of two equivalent methods of accessing the DIINS internal parameters and configuration files: a) via the SHINNADS console and the two way serial communication link to DIINS b) via a separate hand-held or laptop device attached to an available communications port on DIINS.	A.3.3	H & S 1.1.1 a HCI C & D CSC
E	*Data Distribution Component:* This unit shall consist of CCA's capable of distributing the output data at the rates specified in Section 7 of the requirements.	A.3.4	H & S 1.1.2 d Device Drivers CSC
F	*Data Recording Component.*: This unit shall be capable of logging DIINS navigation outputs and other, user-selectable Kalman filter outputs, at rates and durations specified by an operator using either of the control and display methods above. The data to be logged shall be as follows: <u>Raw Sensor Data:</u> • INS • GPS & DGPS	A.3.5	H & S 1.1.4 a Data Collector CSC

Item	Requirement	Parent Item	Component
	• Loran Kalman Filter Outputs: • Filter Residuals • State Vector • Position Estimate • Covariances Miscellaneous • Identified software exceptions • FDI Log text messages • Filter Log text messages		
G	The maximum volume of data that can be logged without changing the removable hardware media shall be: • 1GB on Portable PC hard drive • Typically 4 or 8GB on portable EXABYTE tape drive.	Derived from A.3.5	H & S 1.1.4 b Data Collector CSC
H	The data shall be logged with the format defined in Appendix E (compression may be used) of the System Specification.	Derived from A.3.5	H & S 1.1.4 c Data Collector CSC
I	The DIINS data shall be logged in a sequence of log files, each file covering a two hour time period, with no data gaps between files.	Derived from A.3.5	S 1.1.4 d Data Collector CSC
J	The names of the log files shall be automatically generated by the system, and shall be based on a function of DIINS time.	Derived from A.3.5	S 1.1.4 e Data Collector CSC
K	The logged data shall be able to be replayed at an on-shore analysis facility.	Derived from A.3.5	H & S 1.1.4 f Data Collector CSC

3. Sensor Requirements

A	The system shall have the ability to interface to at least the following sensors: Either: - Dual Sperry Mk-29 Mod 3 Inertial Navigation Systems or - (pending DND procurement approval for the MK-49) - Dual Sperry Mk-49 (NATO SINS) Inertial Navigation System and - Trimble TASMAN (P/Y) GPS receiver. - Leica 12 channel MX94xx series DGPS receivers *Note: The Leica DGPS is located in the SHINNADS system.* - Internav GC500 combined Loran C / GPS(C/A) receiver Or the - Internav LC360 Loran C receiver *Note: It is assumed that DIINS sees the GC500 only as a Loran C system, with outputs identical to the LC360.* *Note: The Internav GC500 is replacing the Internav LC360 Loran-C system in service. DIINS will have to still maintain the capability to process the LC360 in the foreseeable future.* - Differential GPS (DGPS), from a system external to SHINNADS	A.4 C.3	H & S 1.1.2 e Device Drivers CSC
B	1. The allowable DGPS inputs shall be as follows: a. via a modified link with OSL's	C.3	H & S 1.1.2 f

	SHINNADS to allow the DGPS receiver within SHINNADS to provide DGPS data to DIINS; b. via a separate DGPS system that can supply DGPS data in NMEA formats, through an unused serial port on the DIINS ADM.		Device Drivers CSC
C	The hardware interfaces shall be sufficiently general to allow the above sensors to be used in DIINS.	A.4	H
D	The DIINS software, in terms of simple data stream formats, Kalman filter error modelling and Fault Detection, and at the control and display level, must be able to accommodate at least the above sensors.	A.4, C.3	S 1.1.1 b HCI C & D CSC 1.1.3 a Encoders CSC 1.1.5 a Sensor CSC 1.1.7a Prefilter/Filter CSC 1.1.8 a FDI Controller CSC
E	The DIINS ADM shall contain a parameterized sensor library, in which a few of the most common and likely naval INS (specifically the Sperry MK-49 and Litton WSN-5 (the INS used on the TRUMP Class ships)), GPS and other navigation systems can be selected from a list.	C.5	S 1.1.1 c HCI C & D CSC 1.1.5 b Sensor CSC
F	Minimal software changes shall be required to interface to a new sensor. Thus the areas of the code that must be changed to accommodate a new sensor must be kept as separate as possible and documented thoroughly.	C.5	S 1.1.5 c Sensor CSC

4. System Accuracy

A	The system shall be capable of providing navigational accuracies equivalent to those obtained by the XDM, approximately 15-20m (95%) N and E position error and 1-2 arcmin (rms) roll pitch and heading error (Mk-29 integrated with Tasman P/Y GPS).	A.5	S 1.1.7 b Prefilter/Filter CSC

5. Fault Detection and Sensor Redundancy

A	The system shall implement FDIR algorithms equivalent to the Kalman filter residual testing, chi-squared fault detection and multi-filter failure isolation techniques within the DIINS XDM.	A.6	S 1.1.8 b FDI Controller CSC
B	In addition the methodology, as presented in the XDM, of filter selection and reconfiguration under sensor failure conditions, shall be followed.	A.6	S 1.1.8 c FDI Controller CSC

6. Data Distribution Requirements

A	The best integrated estimate of vessel position, speed and heading shall be output on an RS-422 serial line to SHINNADS at least once every 1.28 seconds.	A.7	H & S 1.1.2 g Device Drivers CSC
B	The best estimate of vessel roll, pitch and heading shall be output at least every 40 milliseconds.	A.7	H & S 1.1.2 h

			Device Drivers CSC
C	The contractor shall propose and implement a method for distributing this high rate roll, pitch and heading data (as well as the lower rate position and speed) from DIINS to the rest of the ship. There are 3 distribution requirements: a) Pos/speed/heading and internal DIINS diagnostic info to SHINNADS at ~ 1/second; (Currently met by XDM) b) Pos/speed/heading at ~1/sec for general shipboard system uses; and (Currently met by XDM) c) High rate (25 Hz or greater), low latency (less than or equal to 40 ms) Roll/Pitch/Heading data for weapons or other systems requiring precise, near-continuous attitude information. (New for ADM) *Note: The requirement is to distribute the high rate data at a minimum of 25Hz. The design goal is to distribute the attitude data at a higher rate if possible, e.g. greater than or equal to 100Hz. The high rate data will be achieved by the redistribution of the data received from the INS, but with interpolation and corrections being applied within the ADM based on the DIINS filter outputs.*	A.7 C.4	H & S 1.1.2 i Device Drivers CSC

7. Host Vessel

A	The host vessel will be a Halifax class frigate of the Canadian Navy.	A.8	H

8. Environmental Conditions

A	The DIINS system will be housed within the SHINNADS console on the bridge of the vessel.	A.9	H

9. Reliability

A	The system shall be designed according to generally accepted principles of ruggedized commercial computing systems. No mil-spec requirements are placed on DIINS.	A.10	H
B	The system reliability design target is a minimum of 5000 hours MTBF.	A.10	H

10. Maintainability

A	The system shall be designed to minimize Mean time to Repair (MTTR).	A.11	H
B	Design objectives shall be 15 minutes for first-line (on vessel) repairs. Board level replacements are desirable on the vessel.	A.11	H

11. Human Factors Engineering

A	The system shall be designed to simplify and render most efficient the operator interface.	A.12	H & S
B	The DIINS ADM user shall be able to view and optionally change some of the DIINS	C.2	S

	configuration and filter parameters that are presently set in the DIINS configuration and filter parameter files (SimSetup.cfg and FilterInitx.dat). The parameter values currently in the DIINS XDM parameters files that are not defined below will be setup to appropriate values by the system during initialization. The parameters accessible to the ADM user will be as follows: Configuration Parameters: • Definition of the installed sensor models in DIINS; • Selection of the sensors to be used with DIINS; • Lever Arms for the installed sensors; • RS-232/422 communication channel parameters for the installed sensors Initial sensor measurement values (for filter initialization), including standard deviations and failure and pass fractions and times.		1.1.1 d HCI C & D CSC
C	There shall be two essentially equivalent ways to view and change parameters: a. via a hand held of laptop computer (external control and display unit) interfaced to DIINS through an unused serial port; and b. via a custom screen on the SHINNADS console.	C.2	H & S 1.1.1 e HCI C & D CSC
D	The control and display functions for the DIINS internal parameters, when presented on the SHINNADS console screen shall be consistent with other control and display screens used in SHINNADS.	A.12	S 1.1.1 f HCI C & D CSC
E	The external control and display unit shall present	A.12	S

	Requirement	Derived from	Component
	the similar interface as used when accessing DIINS from the SHINNADS console.		1.1.1 g HCI C & D CSC
F	The operator shall be able to specify the source (if any) of DGPS data as one of the following two sources: a. SHINNADS DGPS Receiver b. Separate DGPS system	C.3	S 1.1.1 h HCI C & D CSC
G	The operator shall be able to select the INS , GPS and other navigation sensors/systems to be used from a list of the available sensors/systems.	C.5	S 1.1.1 i HCI C & D CSC
H	The sensor and Kalman filter configuration interface shall be sufficiently flexible to allow a knowledgeable user the ability to fine tune the parameters of a particular sensor.	C.5	S 1.1.1 j HCI C & D CSC
I	The DIINS ADM (via the SHINNADS display) shall provide the following outputs when operating in the Maintenance Mode: • The Percentage used of the available processing power • The filter currently selected for output • Identified software exceptions • FDI Log text messages • Filter Log text messages	Data previously available via development lab display.	S 1.1.1 k HCI C & D CSC
J	When operating in Maintenance Mode the DIINS software shall allow the operator to inject a hard or soft sensor failure for a selected sensor, which will cause the sensor data to be degraded accordingly. This will be used to verify the correct operation (including failure detection performance) of the DIINS failure detection processing. *Note: This feature can also be used to support acceptance testing of this functionality.*	Derived from A.6	S 1.1.1 l HCI C & D CSC 1.1.5 d Sensor CSC

	Requirement	Derived Weight	Component
K	The DIINS software shall be able to be started without the need to use the HCI to configure the system.	Derived from A.2	1.1.10 a System Config Controller CSC
L	Once the system has been installed and configured, this configuration shall be used as the default configuration.	Derived from A.2	1.1.10 b System Config Controller CSC
M	The operator shall have to take some specific action to allow the configuration control HCI forms to be presented.	Derived from A.2	1.1.1 m HCI C & D CSC

7. LIST OF ACRONYMS

ADM	Advanced Development Model
CCA	Computer Card Assembly
CPU	Central Processing Unit
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
DGPS	Differential Geographic Positioning System
DIINS	Dual Inertial Integrated Navigation System
DND	Department of National Defence
DREO	Defence Research Establishment Ottawa
ECDIS	Electronic Chart and Display Information System
FDI	Failure Detection and Isolation
GPS	Geographic Positioning System
HCI	Human Computer Interface
LAN	Local Area Network
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
NMEA	National Maritime Electronics Association
OSL	Offshore Systems Limited
SCSI	Small Computer System Interface
SDD	Software Design Document
XDM	eXperimental Development Model

APPENDIX A HUMAN COMPUTER INTERFACE FORMS

The following sections briefly describe the Human Computer Interface HCI interface forms presented in this appendix.

The forms presented are primarily for use by the maintenance engineer, with only the form designed for selecting the sensors to be used for the navigation solution being specifically designed for use by the DIINS navigation operator.

Notes:

1. With all of the HCI forms shown in this appendix the data for a particular form is not entered until the "OK" command is entered. If it is required to cancel the data entry for a particular form, the "Cancel" command can be used on the form.

2. The general features of windowing interface HCIs will be available on the DIINS HCI. These will include:

   - Selection and deselection of list items via mouse clicks;
   - Alphabetical finding of list entries by clicking on the list and entering the first letter of the item to be found;
   - Scrolling up and down list items, and their associated data items using the keyboard arrow keys; and
   - Tabbing between the various fields presented on the forms using the keyboard "Tab" key.

3. The GUI prototype forms presented in this appendix were developed using Microsoft Visual Basic for the Windows 95 environment. The OSL SHINNADS PC supports a version of OS/2. Hence, Borland C++ is being used to develop the actual forms for the DIINS ADM. The look of some of the forms may vary from those presented in this appendix. When the C++ forms are available, images of them will be used to replace the forms shown in this appendix.

Figure A-1 Top Level DIINS HCI Form

This form allows the system user to either select the form to be used to select the sensors for the navigation solution, or to select to display the menu of options available for maintaining DIINS.

Figure A-2 Sensor Selection Form

This form allows the operator to select which sensors will be used to generate the navigation solution using DIINS. Only the sensors previously physically installed and installed in the

software using the Sensor Installation Form (Figure A-4) show up as being available for selection for the navigation solution.

Figure A-3 DIINS Maintenance Mode Menu Selections

This form is presented when "Maintenance" is selected on the top-level-form (Figure A-1). If "OperationalMode" is selected the form display reverts back to the top level form again (Figure A-1). The following options are available for selection on this form, and when selected cause the display of the associated lower level forms to enable the operator to review and modify various system parameters:

- Select Sensor
- Operational Mode
- Install Sensors
- Lever Arms
- Communications Setup
- FDI Parameters
- Logging Setup
- Failure Inject
- System Status

The prototype HCI also includes an alternative look-and-feel form for the entry of Lever Arm and Communications Setup information.

Figure A-4 Sensor Installation Form

This form presents the user with a list (on the left) of the available sensor models programmed into the DIINS software. The user (normally the maintenance engineer) selects and installs the sensors using this form to comply with the sensor configuration that is physically wired to the DIINS hardware. If the user makes a mistake or the sensor becomes unavailable, the installed sensor can also be removed using this form. This form also allows the user to set or clear a check box which controls the distribution of the high rate attitude information from DIINS to the other ship systems.

Figure A-5 Sensor Lever Arm Setup Form

Before using DIINS to provide a navigation solution it is necessary to inform DIINS of the locations of the sensors relative to the reference point used within DIINS. Using this form the operator selects the sensor for which data is to be reviewed/modified and enters new data as required.

For the DIINS ADM, the Reference point equals the Center of Floatation (CoF):

"Xc: distance FORWARD of reference point (negative for Aft)"
"Yc: distance ABOVE reference point(negative for below)"
"Zc: distance STARBOARD of reference point (negative for Port)"

The software would then have to translate these to DIINS coordinates using the Equations A-1, where Xd, Yd and Zd represent the DIINS coordinates:

$$Xd = Xc \quad \text{(Eqns A-1)}$$
$$Yd = Zc$$
$$Zd = -Yc$$

Note:

A feature that could be considered for addition, provided it is in keeping with project budgets and time scales, would be the ability to change the reference point. For example once the data is entered, the default reference point could be changed to INS1, and the lever arms would be recalculated automatically. Note that the DIINS output is always referenced to the (0,0,0) of the coordinate frame, so if (0,0,0) is the CoF, then that is the reference point. If (0,0,0) is INS1, then the outputs are all referenced to INS1, and so on.

Figure A-6 Sensor Communication Channel Setup Form

DIINS communicates with SHINNADS and the other ship systems using serial communication interfaces, e.g. RS-232/422. This form allows the user to review and modify the parameters associated with the communications channel for the selected sensor. The available channel options are presented via drop down combo boxes, minimizing the data entry/checking requirements.

Figure A-7 FDI Parameters Setup Form

This form allows the system user to set up various initial values to be used by the FDI algorithms to detect hard and soft failures.

Figure A-8 Data Logging Setup Form

This forms allows the user to select the various data items to be logged during the operation of DIINS. A check box is provided for each sensor to select whether the sensor is to be logged or not. If the sensor is not to be logged, its Log Interval and Nominal Bytes/Second display fields are set to zero. A list of the available filters is provided to allow the user to select which filter outputs should be logged.

The form indicates the number of bytes per day that would be logged with the current form selections. This enables the user to determine how long the available storage volume for logging will last, or to be able to select the sensor and filter data to be logged to approximate a specific bytes/day requirement. There are two logging rate scale factors available on the form. The "Overall Rate Scale Factor" (bottom left) allows the user to adjust the overall logging rate by increasing the period at which the data is logged. Settings of the Overall Rate Scale Factor are 1.0 (Nominal value), 0.5, 0.25 and 0.1 (i.e. log at 1/10 th the rate of the nominal logging). In addition to the overall rate scale factor, each sensor has an individual "Rate Scale" factor that can be used to override the overall scale factor for the selected sensor. If the individual Rate Scales have been set and the Overall Rate Scale Factor is then modified, this resets all of the individual Rate Scale factors to the overall scale factor again. The form also displays (middle right) the number of bytes per second currently being logged for the filters with the current filter selections. The period at which the filter is updated is also displayed on the lower right of the form.

Figure A-9 Failure Injection Form

This form enables the DIINS sensor failure detection mechanisms to be verified to be operational. Both hard (complete failure of the sensor) and soft (small deviations in the sensor outputs due to sensor failure/degradation) failures can be injected, and the failures can be timed to occur either at a particular time (absolute timing) or relative to the current time. The duration of the injected failure can also be programmed via the form. The failure injection capabilities will be prohibited when the system is in operational mode, i.e. when the form of Figure A-1 is displayed or when the same form is displayed in icon style.

Figure A-10 System Status Form

This form enables the DIINS user to monitor the load currently being placed on the DIINS processor by the processing taking place in the current system configuration, e.g. which may include the operation of data logging. The form indicates which filter is currently selected by the DIINS FDI for provision of the navigation solution. The form also displays a dynamically updated list of software exception events, FDI Log and Filter Log text messages. Note, FDI and Filter logged data in general will not be displayed in the event list on this form. Only specific text messages that will be included in the logged data to support the monitoring of the correct operation of the FDI and Filter processing functions in general will be displayed. During nominal system operation no software exception events should be shown in the list. The exceptions raised by the software may be due to either software or hardware failures. The list of exceptions and log events can be cleared by the user.

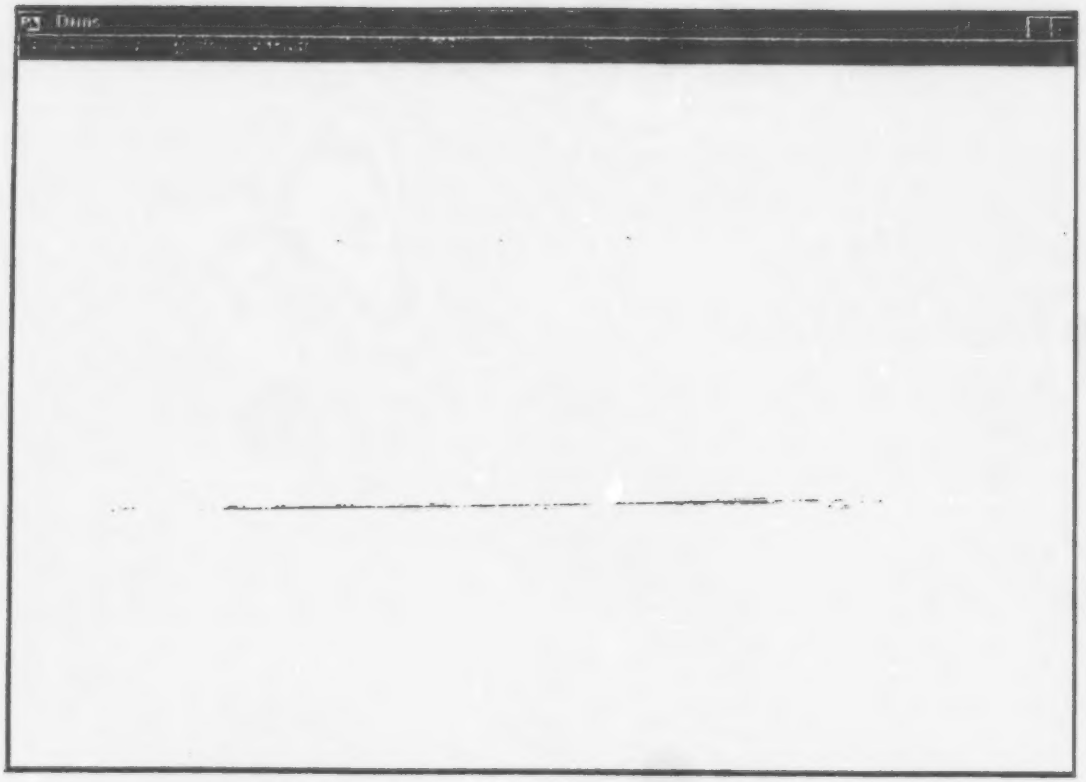


Figure A-1 Top Level DIINS HCI Form

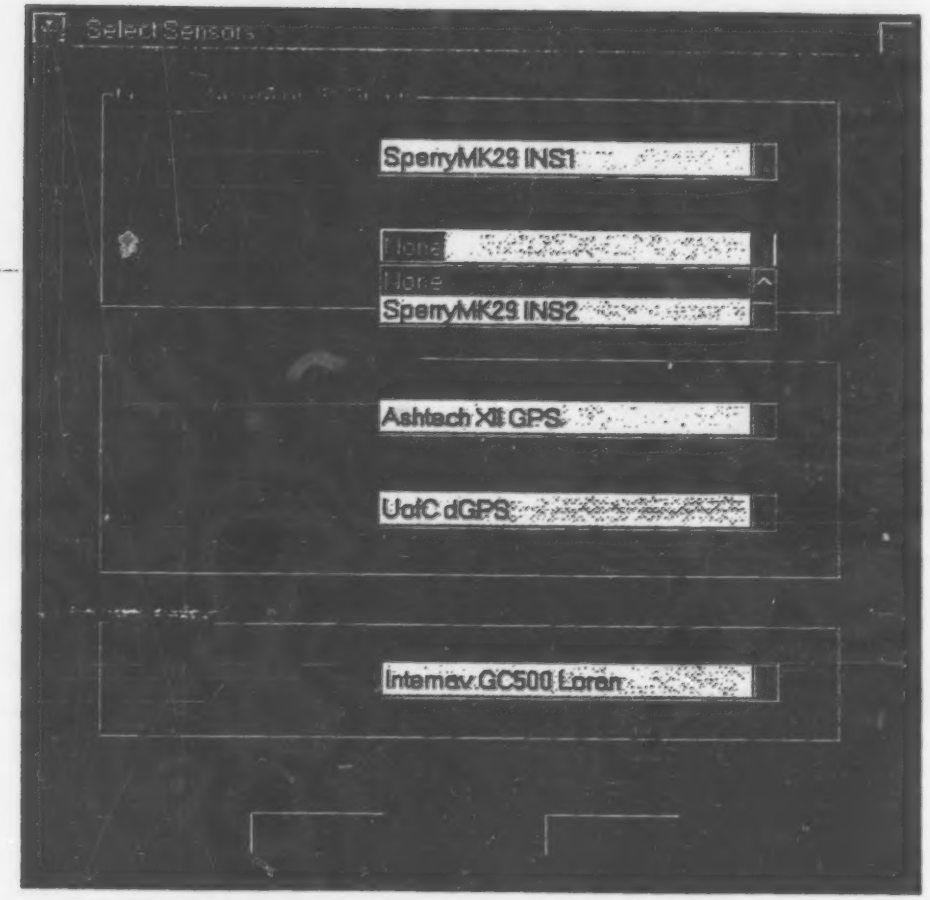


Figure A-2 Sensor Selection Form (from installed sensors)

Figure A-3 DIINS Maintenance Mode Menu Selections

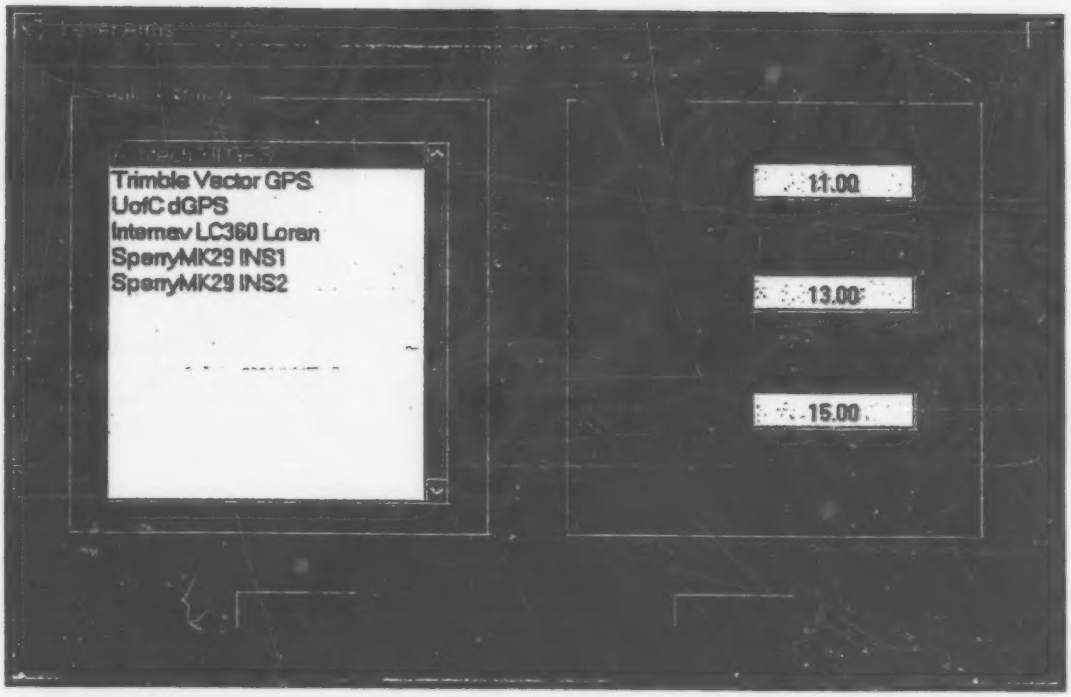


Figure A-5 Sensor Lever Arm Setup Form

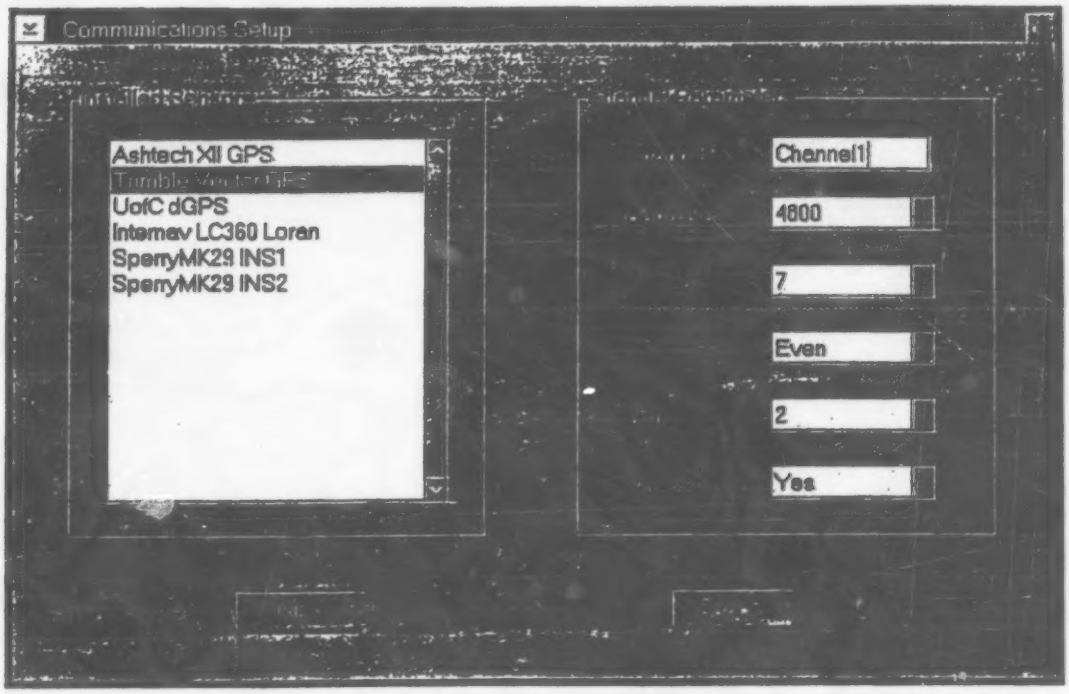


Figure A-6 Sensor Communication Channel Setup Form

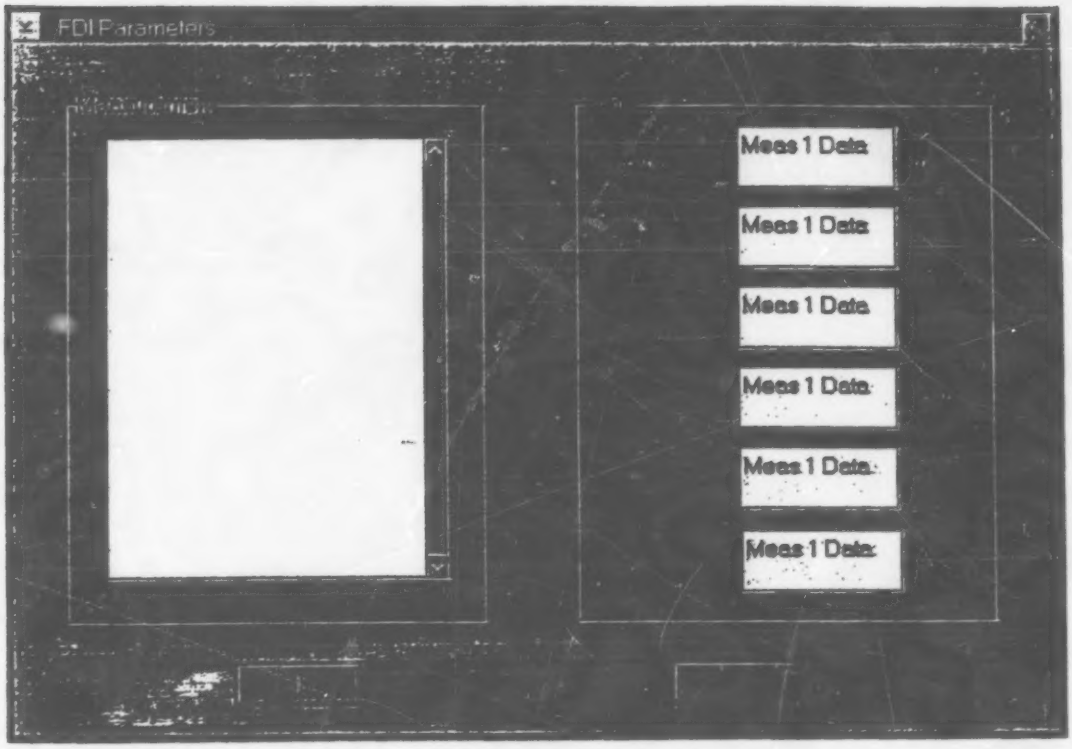


Figure A-7 FDI Parameters Setup Form

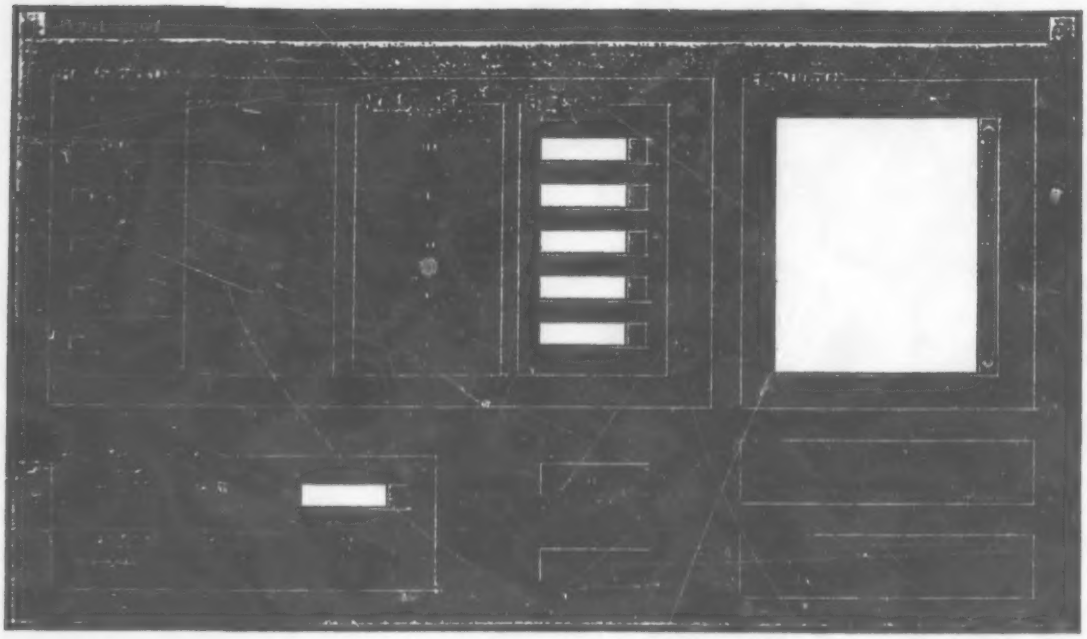


Figure A-8 Data Logging Setup Form

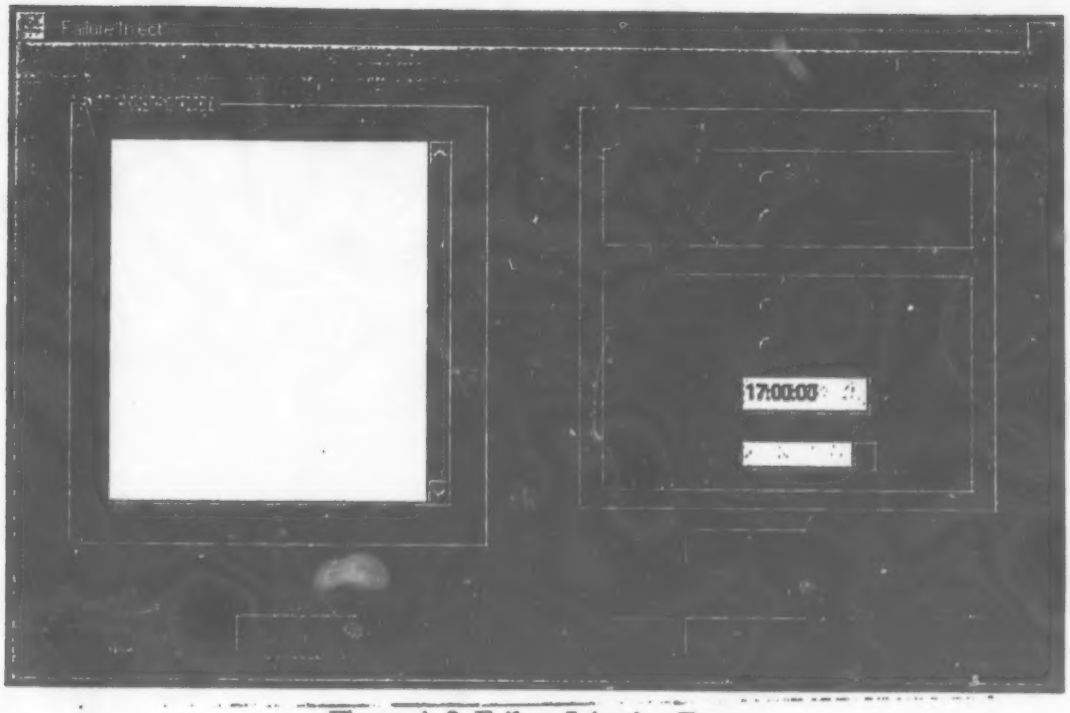


Figure A-9 Failure Injection Form

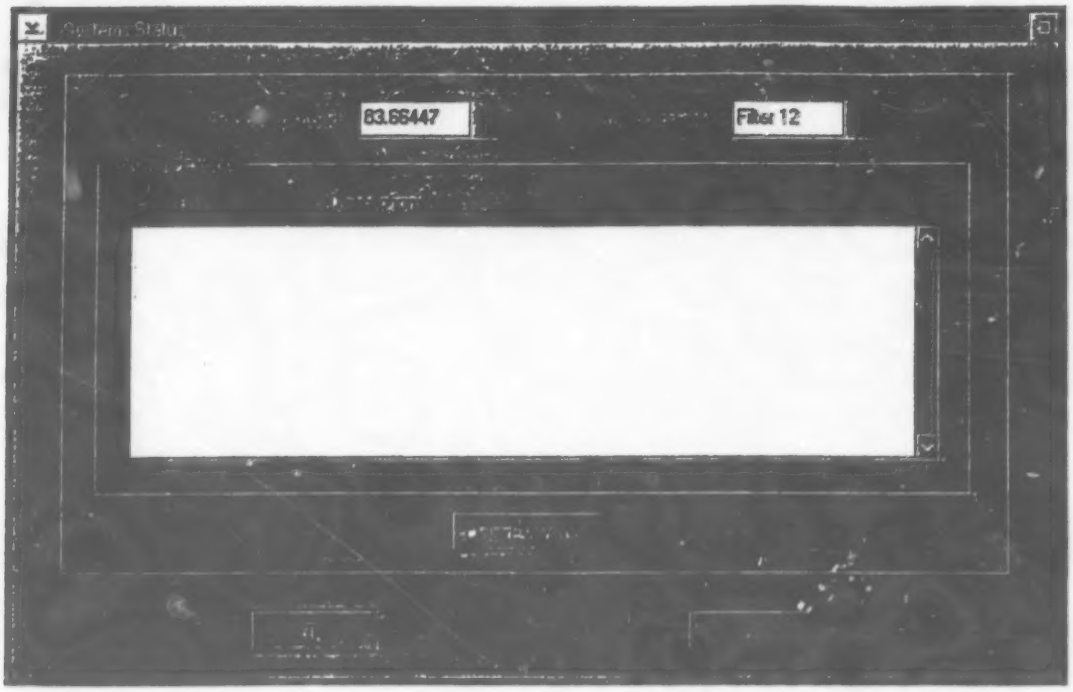


Figure A-10 System Status Form

APPENDIX B DIINS ADM SOFTWARE OPERATION

B.1 Introduction

This appendix presents sequence diagrams of the ADM operation generated using the Rational Rose CASE tool. These diagrams represent the software operations that take place between the significant Computer Software Components (CSCs) in the ADM.

In order to produce the model contained in this appendix, certain decisions have been made as to how the sensor specific configuration data within the ADM will be organized. The proposed configuration data solution is presented in Appendix F.

This model used took as its starting point the known behavior of the DIINS XDM system. The initial XDM model was extended to include, in particular, the new HCI features to be provided in the ADM to enable the user (system operator or maintainer) to more readily configure and monitor the operation of DIINS when in port or at sea. The behavioral description defined in the sequence diagrams presented in this appendix should be read in conjunction with the relevant HCI interface forms defined in Appendix A, and the SHINNADS to DIINS data transfer NMEA formatted messages defined in Appendix E. The content of these three appendices (A, C and E) fully define the high level SHINNADS to DIINS ADM interface operational behavior and associated timing and formats.

The diagrams presented in this appendix have been simplified to some extent. They do not show the details of all of the interactions that take place. In particular, the NMEA encoder and decoder objects are not shown explicitly, but the use of these objects is implied when two configuration items that communicate via NMEA format are shown interacting on the diagrams, e.g. the HCI receiving information from the Data Collector or sending information to the SystemConfigController. In addition, the details of MotherOfAllInterruptHandlers and the PortHandlers have been omitted from the ADM diagrams in order to leave more room for other objects.

There are two main points of contact between the DIINS HCI and the DIINS Adm software. These points are the Data Collection CSC (DIINS->HCI) and the SystemConfigController (HCI->DIINS). The transfer of data between the HCI and these two CSCs is via the NMEA format defined in Appendix E. Appendix C presents sequence diagrams defining the detailed behavior of the HCI interface control software.

B.2 DIINS ADM Use Cases

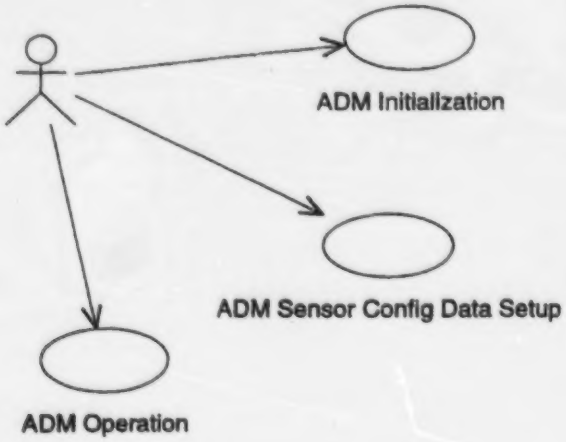


Figure B.2-1 DIINS ADM Use Case Model

F2

B.3 DIINS ADM Initialization

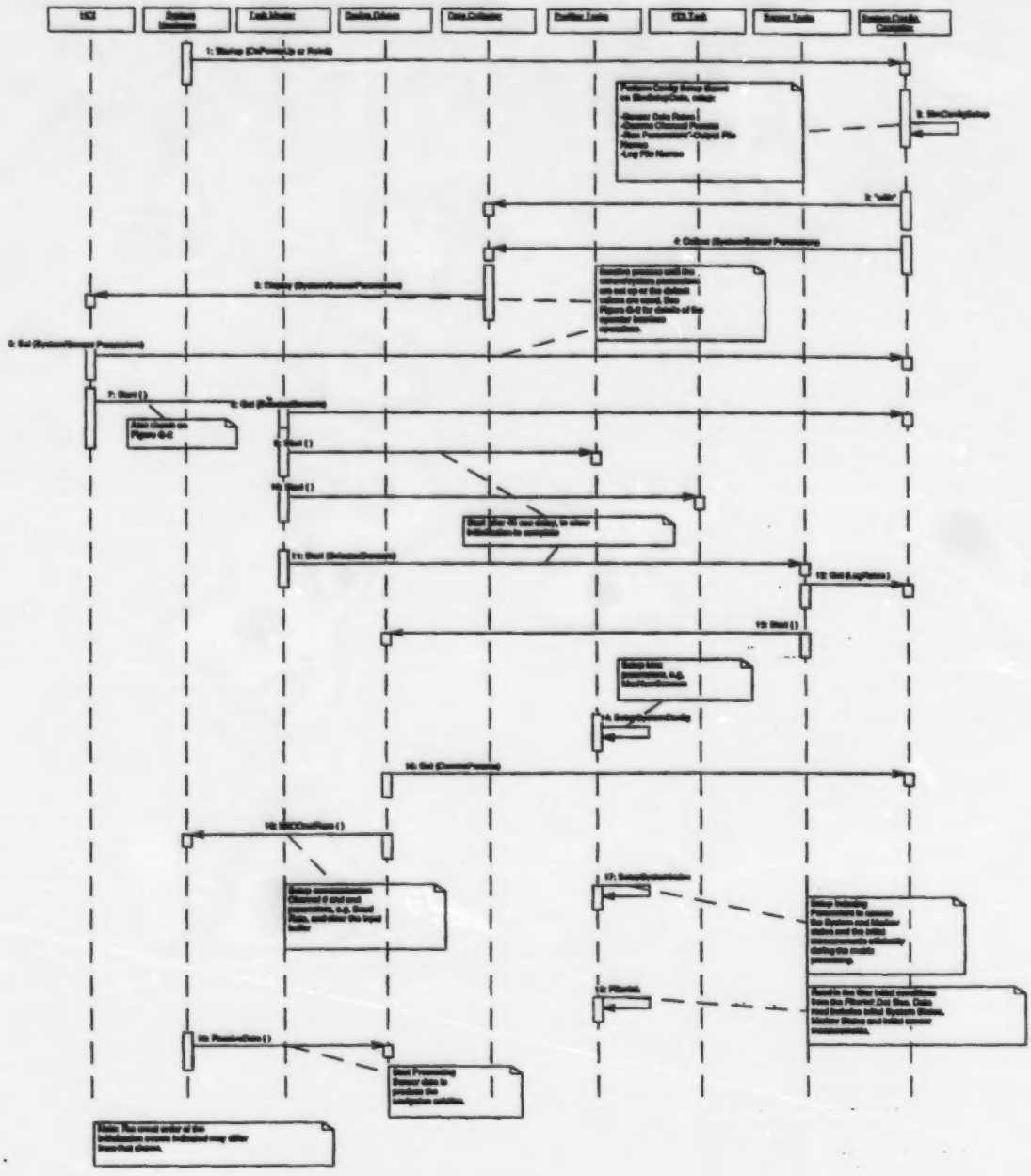


Figure B.3-1 DIINS ADM Initialization

B.4 DIINS ADM Sensor Configuration Data Setup

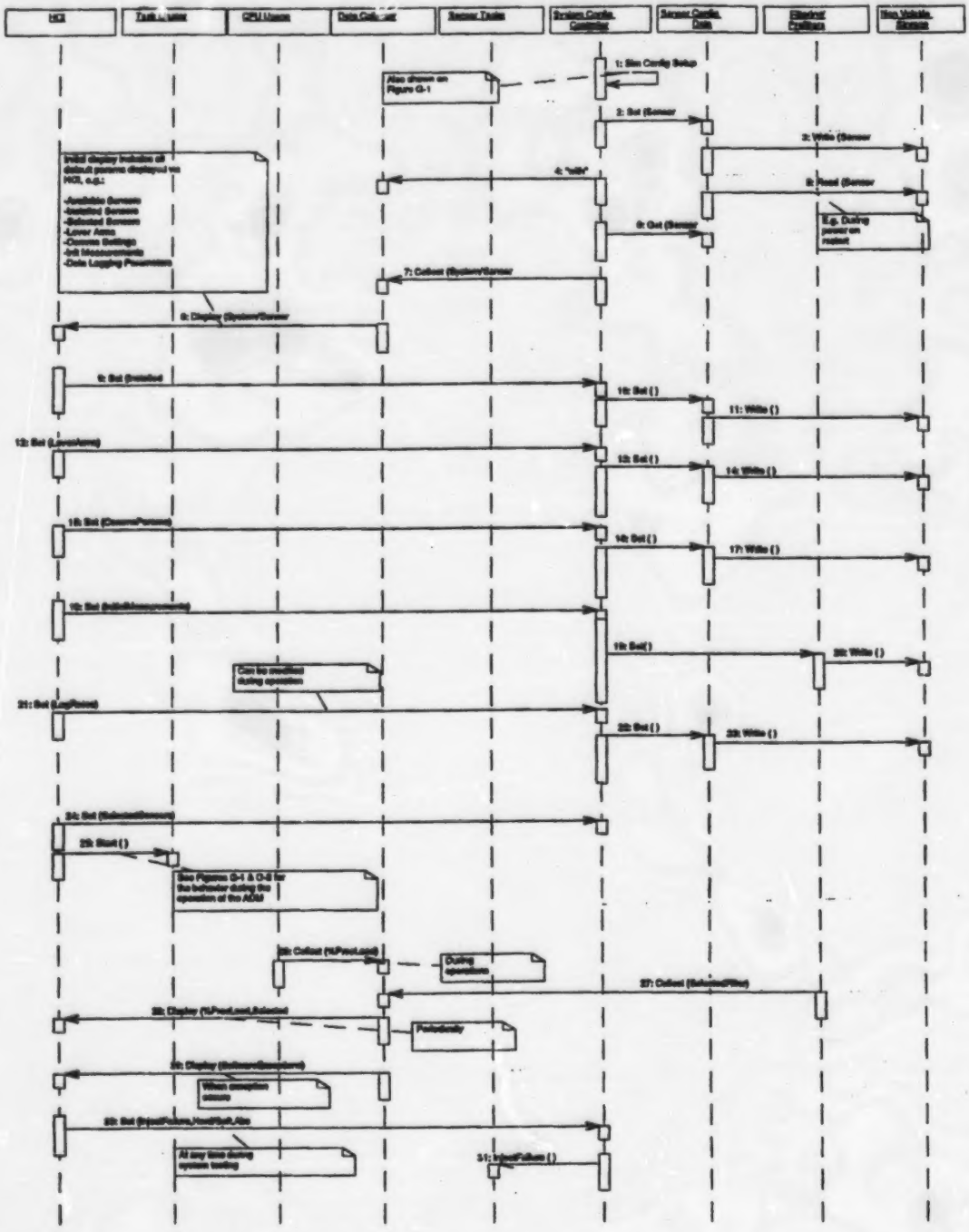


Figure B.4-1 DIINS ADM Sensor Configuration Data Setup

B.5 DIINS ADM Operation

In Figure B-4 the OSL Drivers are shown; whereas, the new Data Collection CSC is also involved in the transfer of the data, as defined in the body of this document.

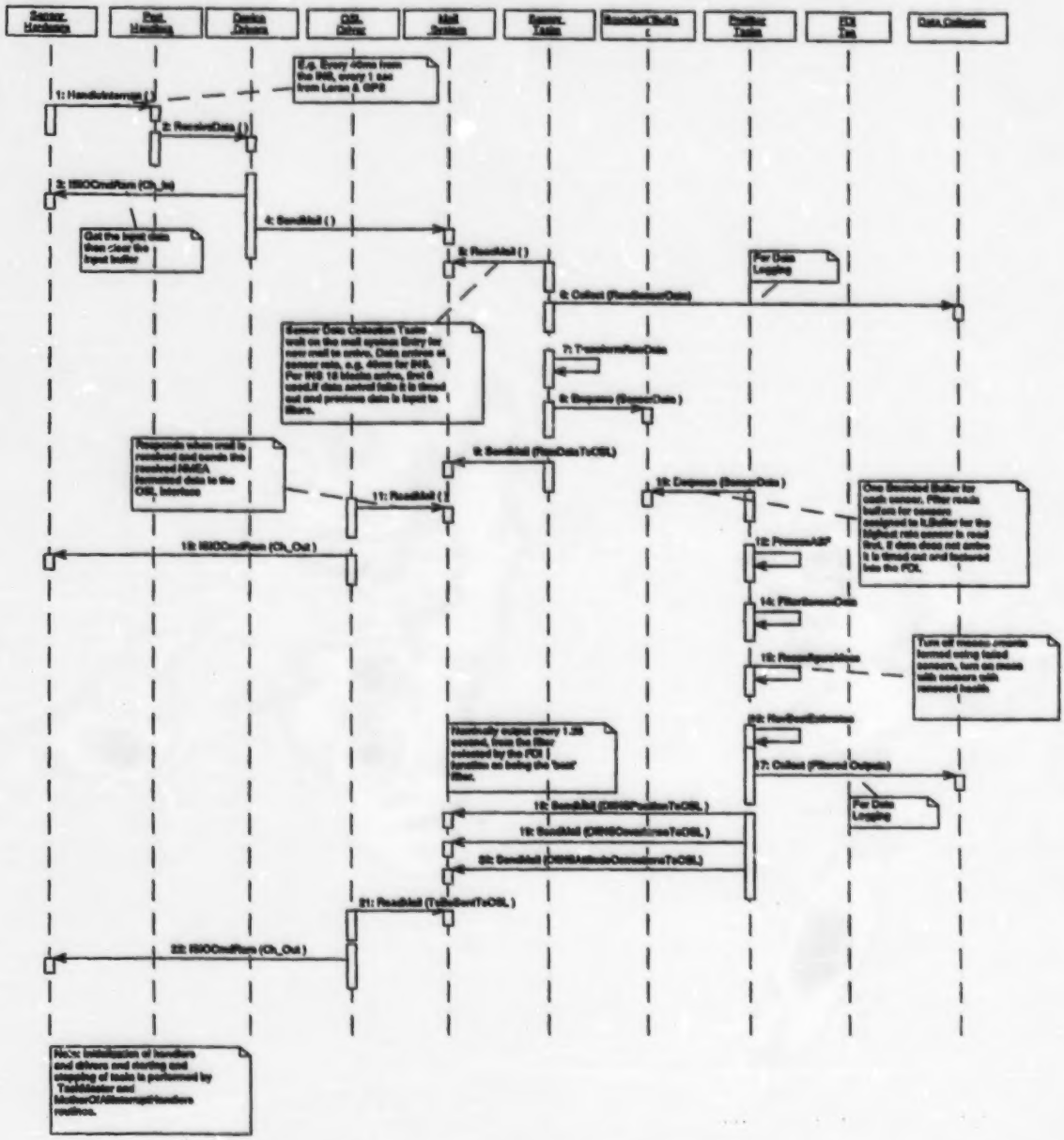


Figure B.5-1 DIINS ADM Operation

APPENDIX C DIINS HCI CONTROL AND DISPLAY SOFTWARE DESIGN

C.1 Introduction

This appendix contains the details of a software design model generated using the Rational/Rose CASE tool. This model uses the UML Object-Oriented design methodology. The diagrams in this appendix have all been extracted from this tool. The model is to be used to generate automatically the framework of the C++ code to be used to implement the HCI functionality. The tool automatically generates skeleton code for the packages, classes, methods and attributes defined in the model. Once the initial code is generated, it is then fleshed out using an editor, the Borland C++ environment in this case, and compiled and tested as with traditional coding techniques. The code, at any time, can be fed back into the CASE tool to update the model held in the tool. Further modifications to the model can then take place in the CASE tool before generating the next version of the code, which can then be compiled and tested again. This iterative approach to software development using this tool is known as round-trip engineering.

C.2 DIINS HCI Control and Display CSC Packages

Figure C.2-1 illustrates the overall software package that the user interacts with.

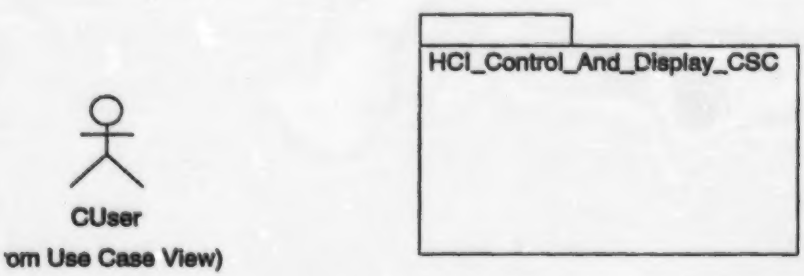


Figure C.2-1 HCI Control And Display Package

The HCI Control And Display CSC is further divided into the CSCs shown in Figure C.2-2

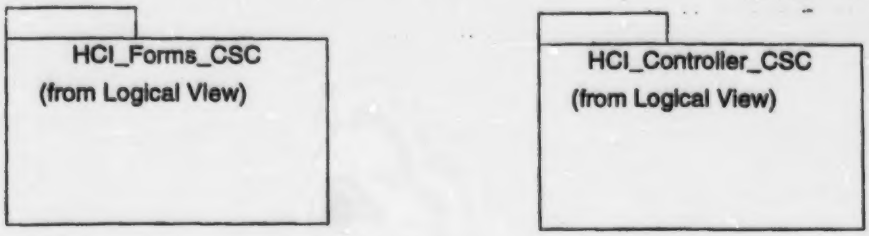


Figure C.2-2 HCI Control And Display CSC Sub-Packages

Figure C.2-3 illustrates the software components in terms of lowest level packages that contain the classes used to implement the functionality and the Main Program that is used to instantiate objects of all of the classes and to provide the overall logic to create the behavior of the system to satisfy the requirements. In Figure C.2-3, the Forms package is allocated to the HCI_Forms_CSC of Figure C.2-2, and the other packages are allocated to the HCI_Controller_CSC.

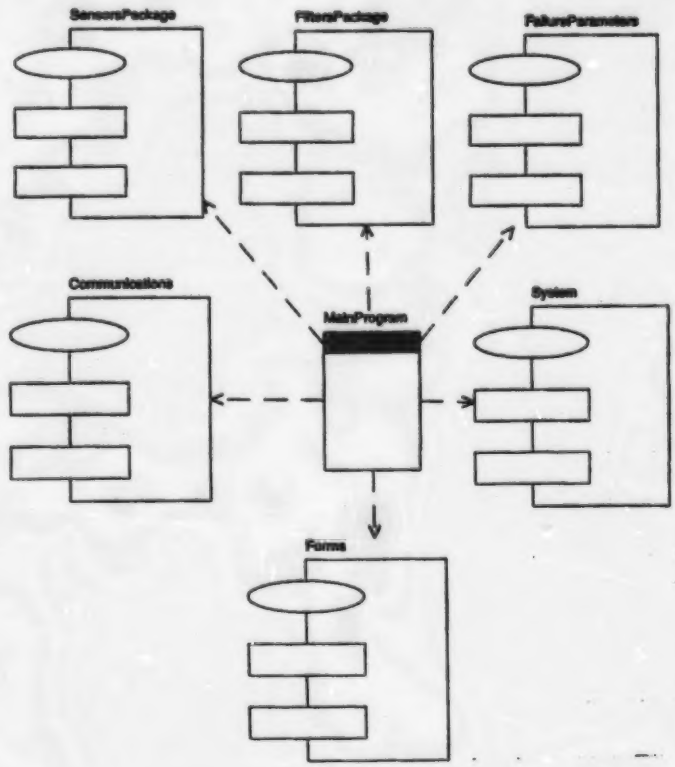


Figure C.2-3 Software Component Packages

C.3 DIINS HCI Control and Display CSC Classes

C.3.1 Forms Package Classes

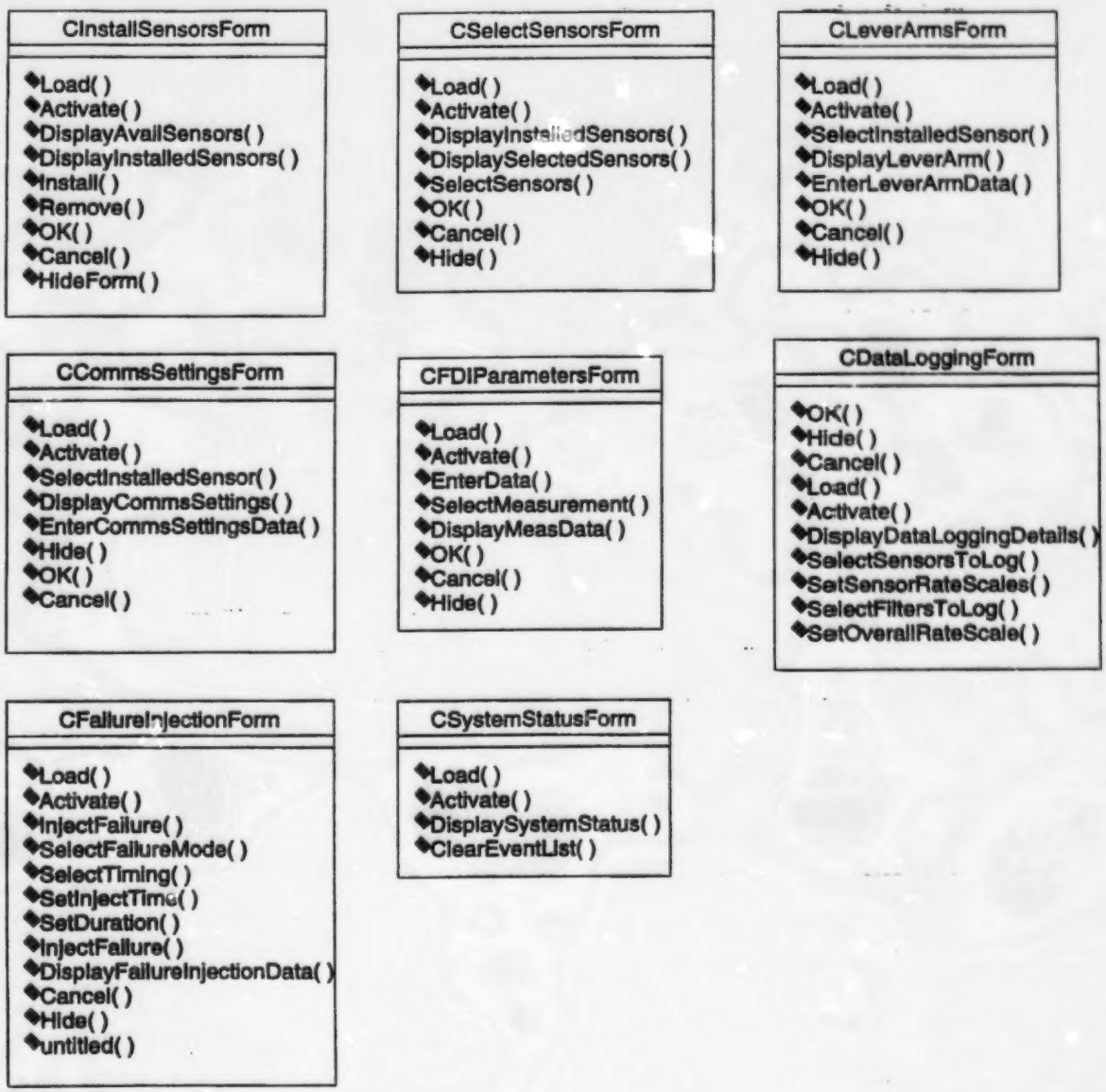


Figure C.3.1-1 Forms Package Classes

C.3.2 Sensors Package Classes

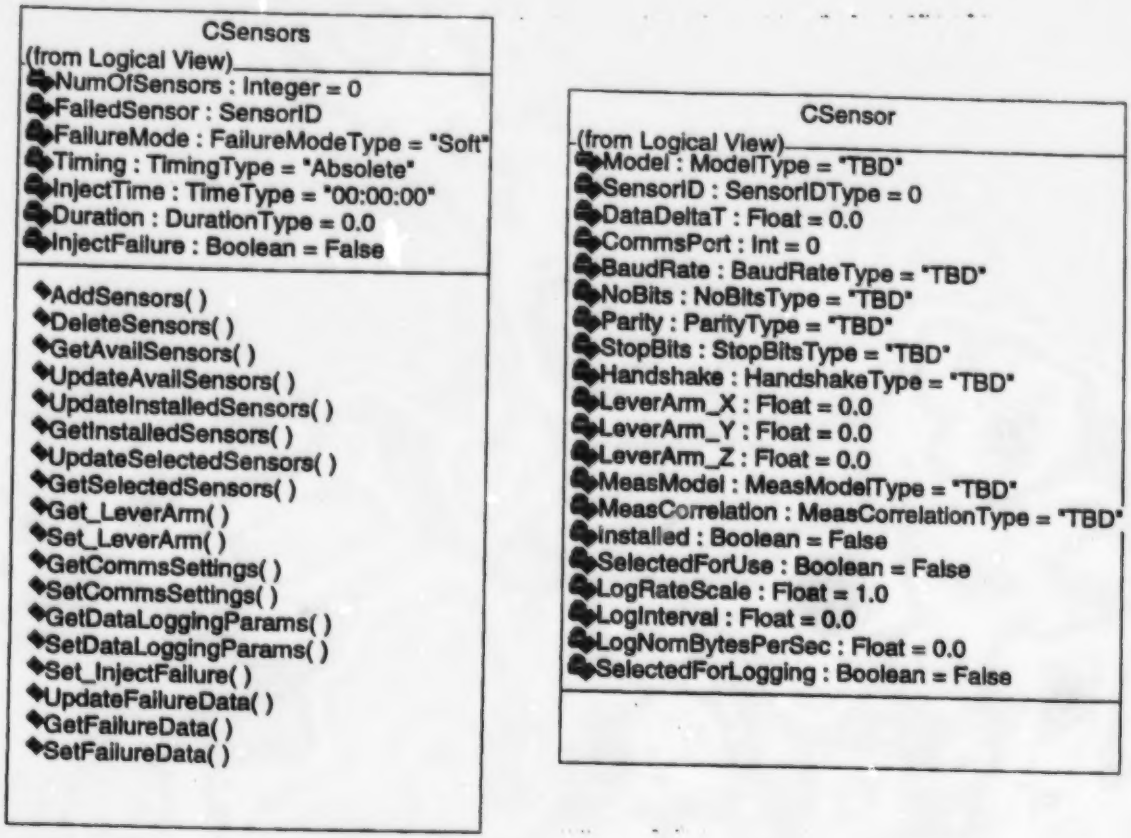


Figure C.3.2-1 Sensors Package Classes

C.3.3 Filters Package Classes

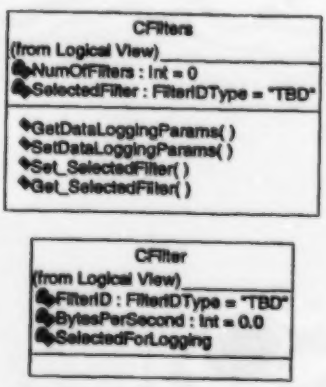


Figure C.3.3-1 Filters Package Classes

C.3.4 Failure Parameters Package Classes

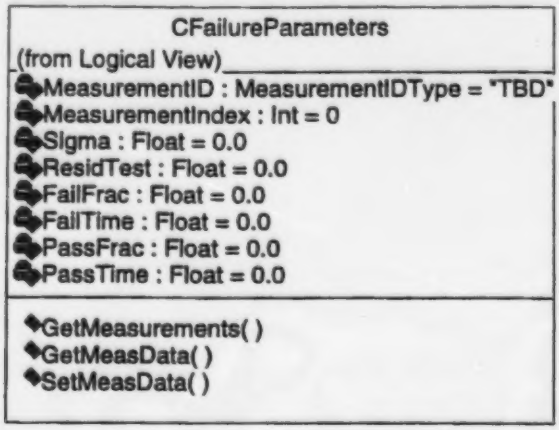


Figure C.3.4-1 Failure Parameters Package Classes

C.3.5 Communications Package Classes

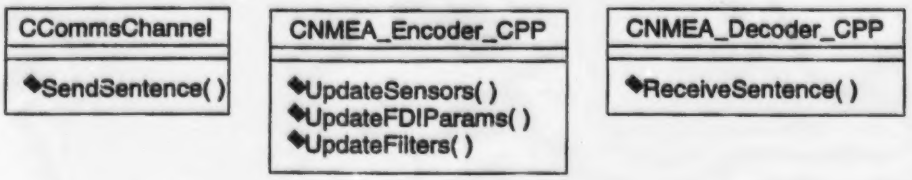


Figure C.3.5-1 Communications Package Classes

C.3.6 System Package Classes

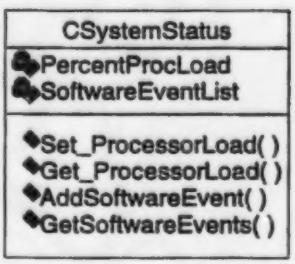


Figure C.3.6-1 System Package Classes

C.4 System Behavior Sequence Diagrams

C.4.1 Sensor Installation Sequence

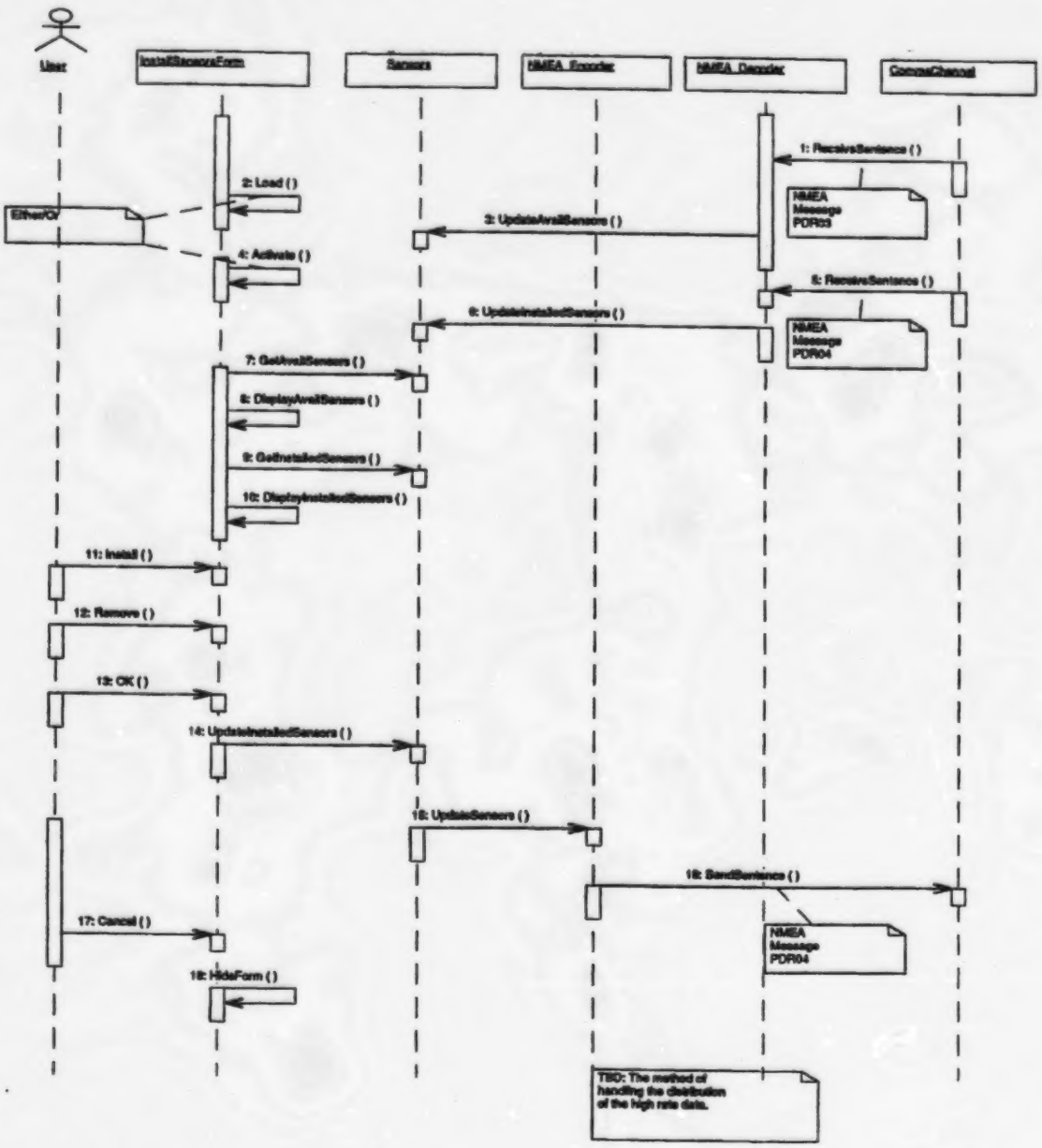


Figure C.4.1-1 Sensor Installation Sequence Diagram

C.4.2 Select Sensors Sequence

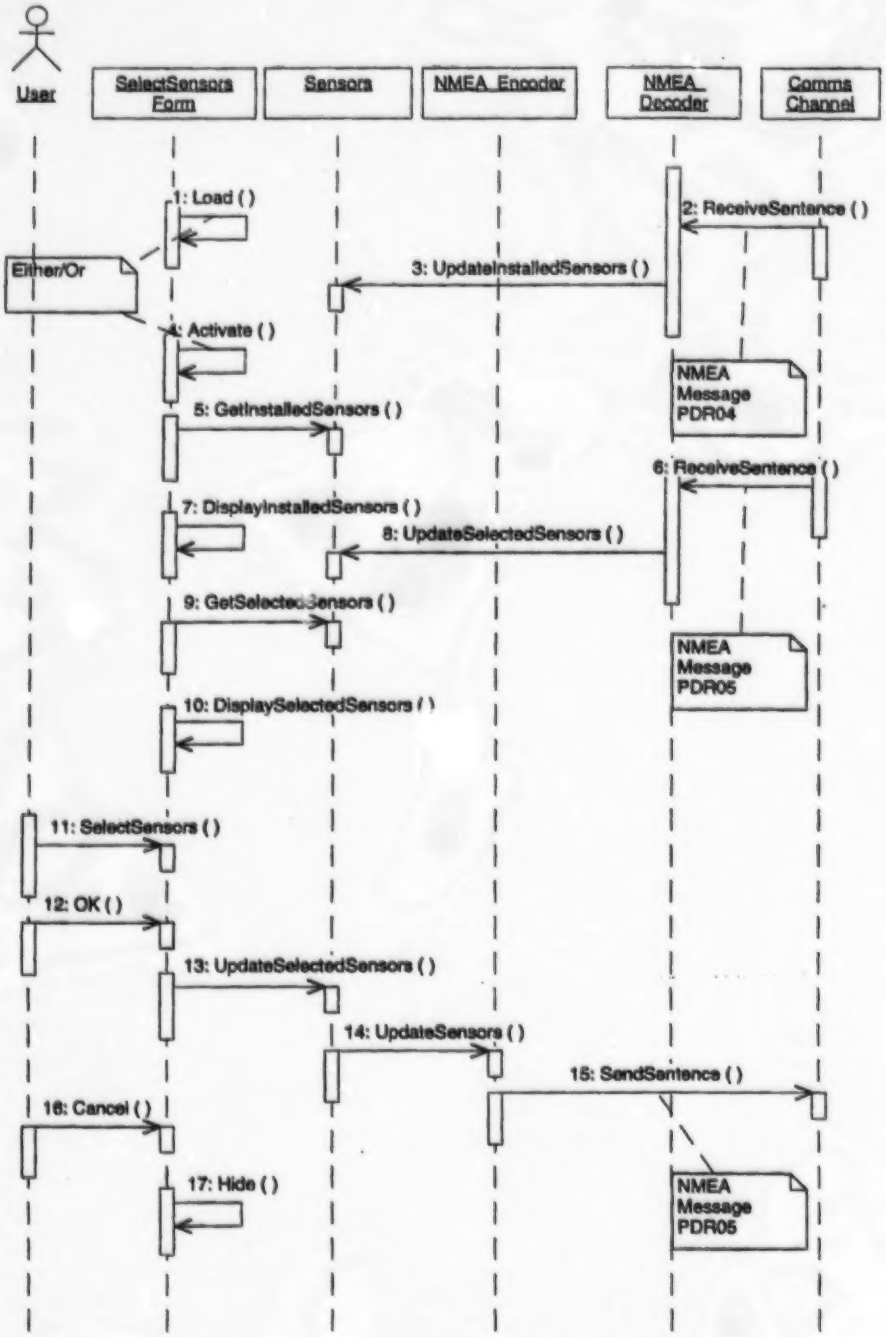


Figure C.4.2-1 Select Sensors Sequence Diagram

C.4.3 Lever Arm Setup Sequence

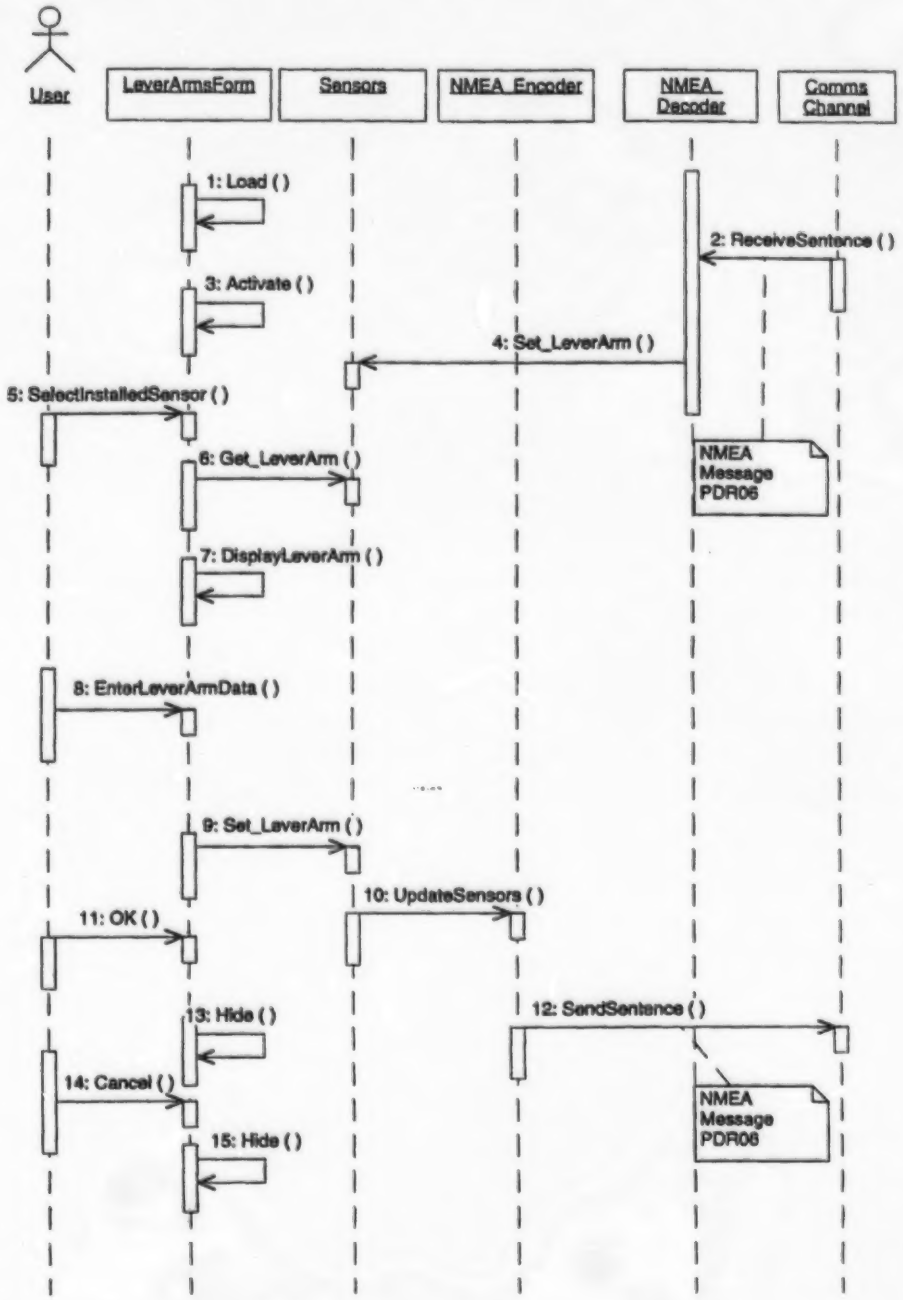


Figure C.4.3-1 Lever Arm Setup Sequence Diagram

C.4.4 Communication Parameters Setup Sequence

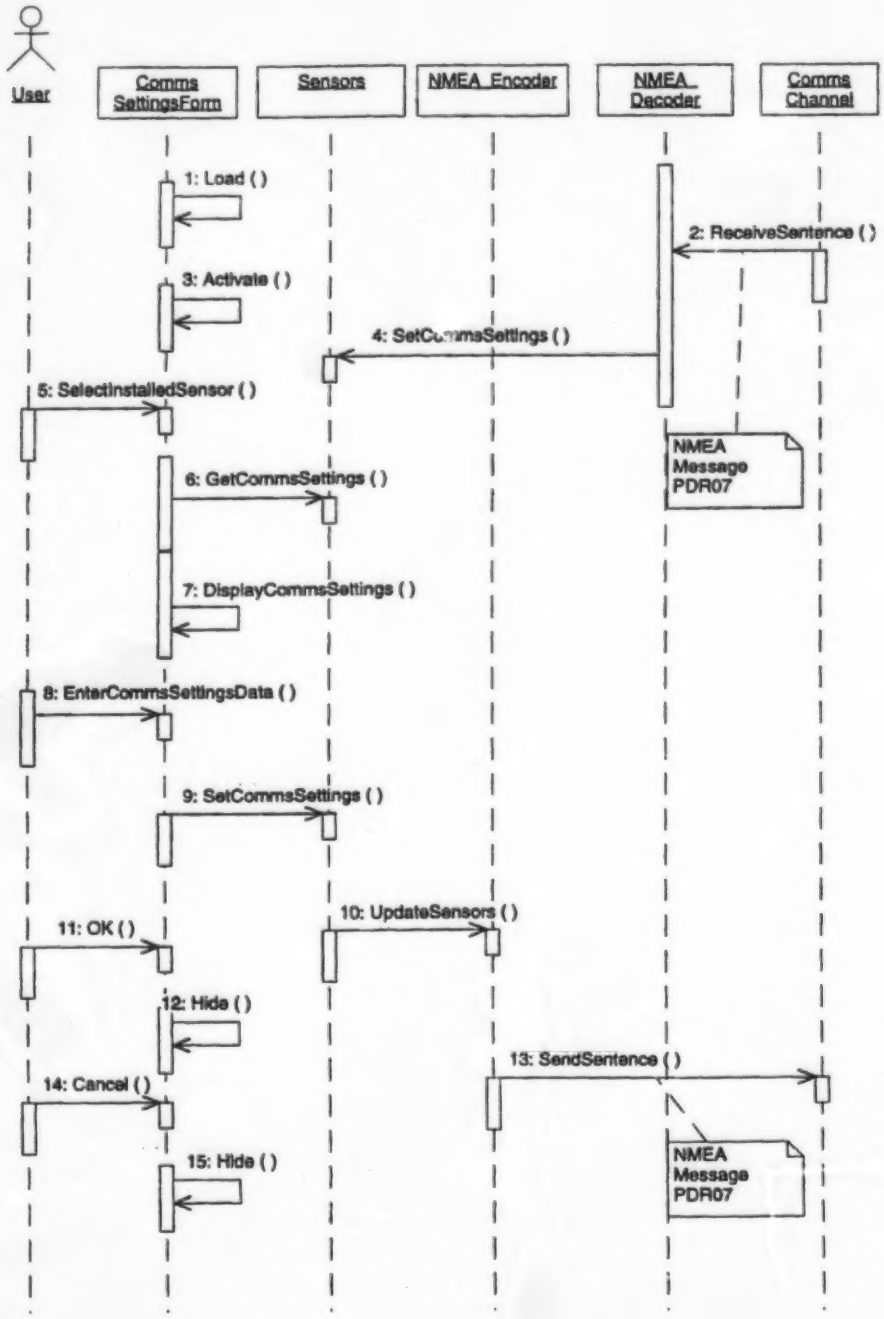


Figure C.4.4-1 Communications Parameters Setup Sequence Diagram

C.4.5 FDI Parameters Setup Sequence

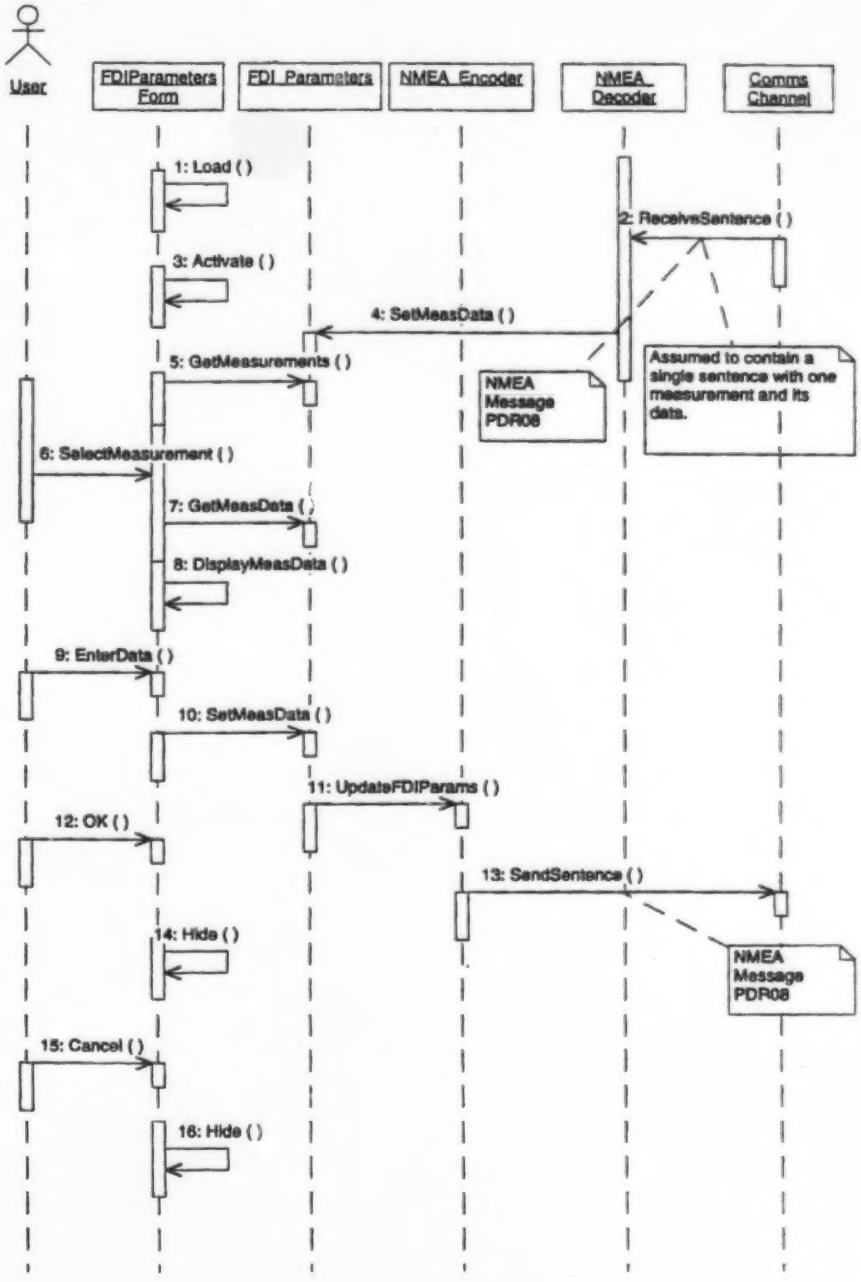


Figure C.4.5-1 FDI Parameters Setup Sequence Diagram

C.4.6 Data Logging Setup Sequence

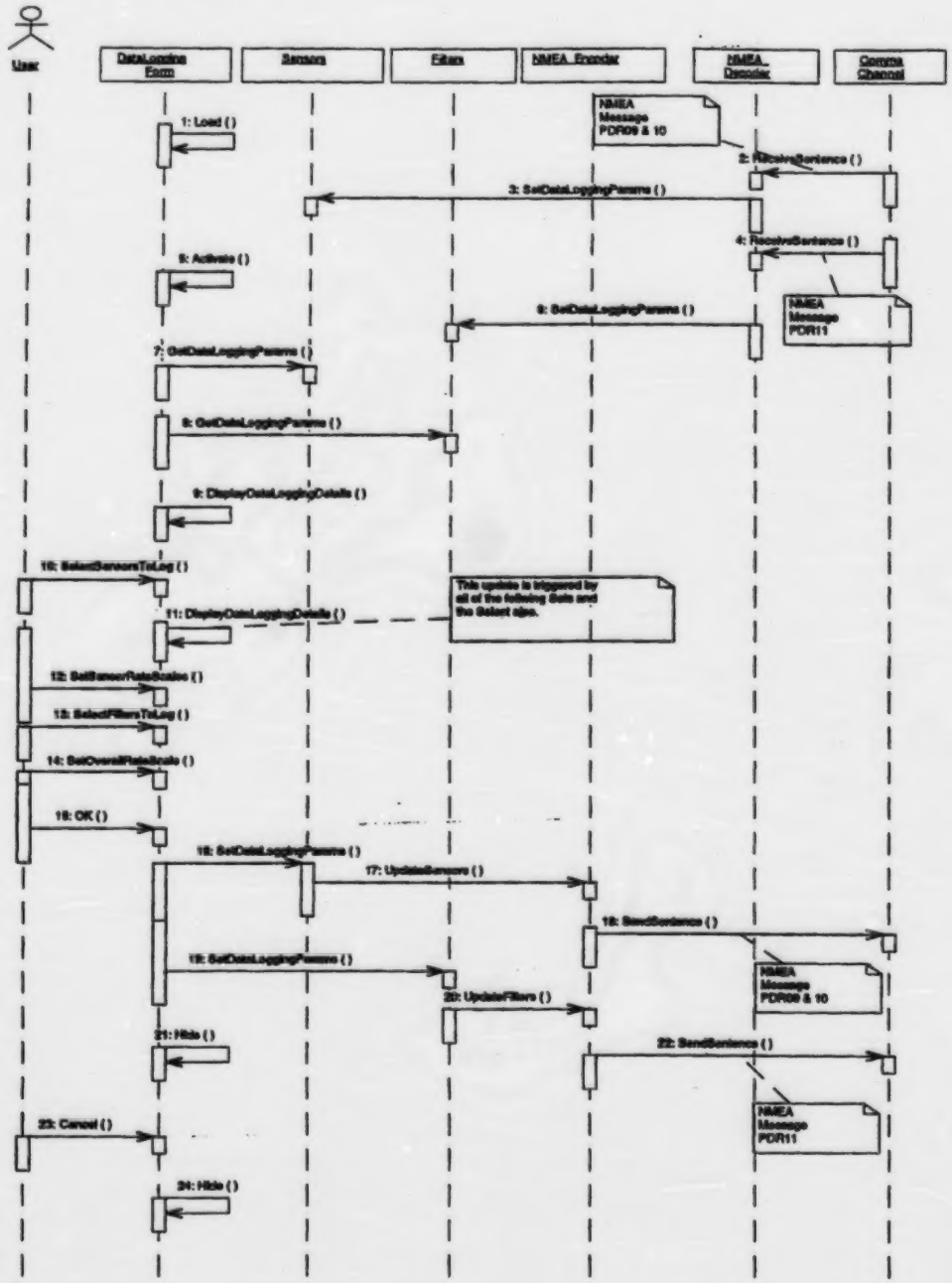


Figure C.4.6-1 Data Logging Setup Sequence Diagram

C.4.7 Failure Injection Sequence

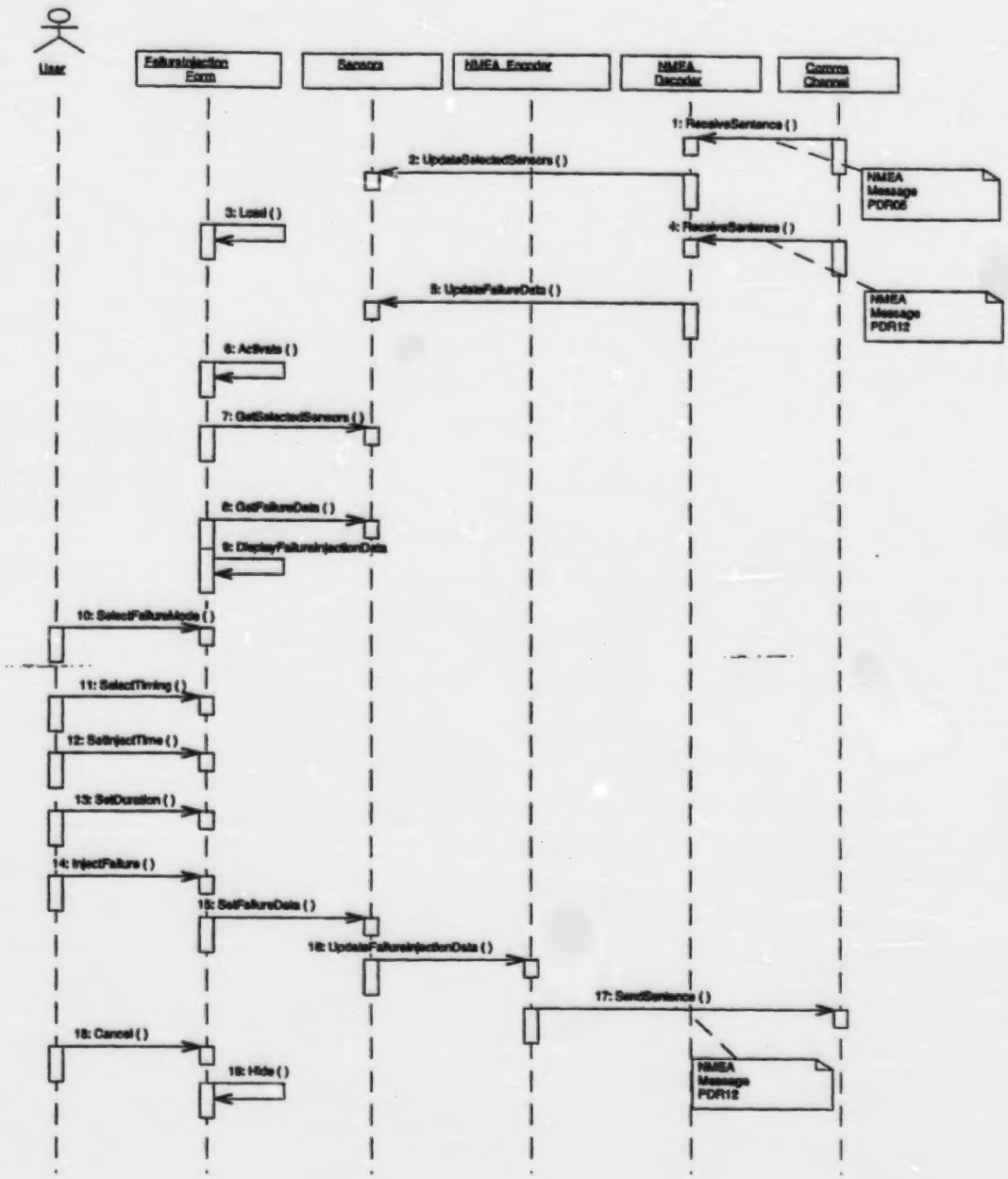


Figure C.4.7-1 Failure Injection Sequence Diagram

C.4.8 System Status Monitor Sequence

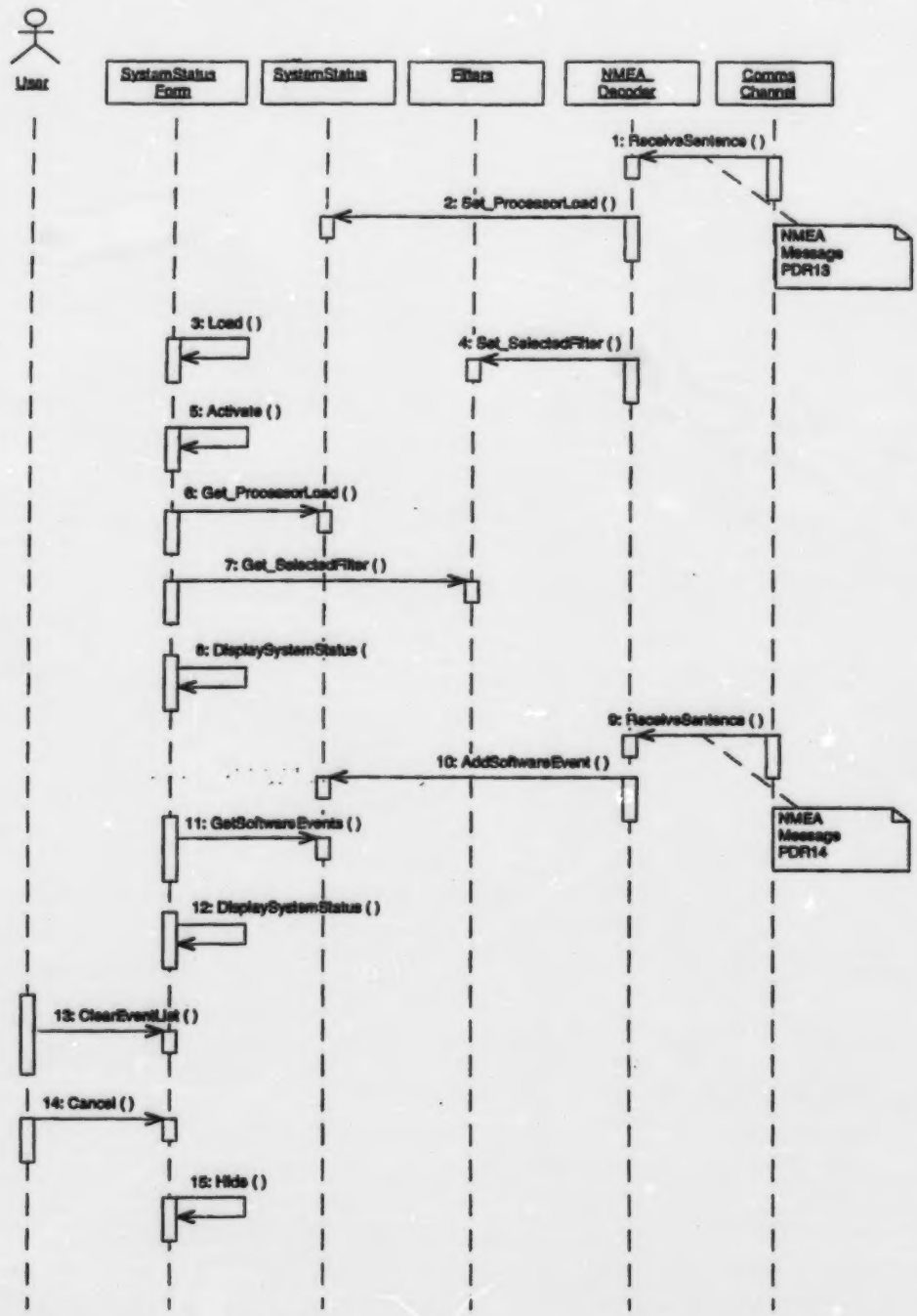


Figure C.4.8-1 System Status Monitor Sequence Diagram

APPENDIX D DATA LOGGING REQUIREMENTS

This appendix indicates (based on the data logging performed in the DIINS XDM) the nominal number of bytes per second required to be logged in the DIINS ADM. This data logging requirement represents a maximum to the amount of data that can be logged from the DIINS ADM. In practice the system user will be able to vary the volume of data logged using the data logging options available via the DIINS Human-Computer Interface Data Logging form.

The DIINS DataLogger logs both Raw Sensor and Filter data. The following sections indicate the format in which it is planned to log the various data items within the DIINS ADM.

Raw Data breakdown is as follows:

. GPS (NMEA) per 1.0 sec
"$R2,<DIINS time><Length of ZDA data><data><CR><LF>"
=> 4 + 8 + 2 + 37 + 2 = 53
"$R2,<DIINS time><Length of GGA data><data><CR><LF>"
=> 4 + 8 + 2 + 66 + 2 = 82
"$R2,<DIINS time><Length of VTG data><data><CR><LF>"
=> 4 + 8 + 2 + 38 + 2 = 54

==========> 189 bytes per 1.0 sec

. dGPS (NMEA) per 1.0 sec
"$R4,<DIINS time><Length of ZDA data><data><CR><LF>"
=> 4 + 8 + 2 + 37 + 2 = 53
"$R4,<DIINS time><Length of GGA data><data><CR><LF>"
=> 4 + 8 + 2 + 66 + 2 = 82
"$R4,<DIINS time><Length of VTG data><data><CR><LF>"
=> 4 + 8 + 2 + 38 + 2 = 54

==========> 189 bytes per 1.0 sec

. INS1 per 1.28 sec
16 X "$R0,<DIINS time><Length of data><data><CR><LF>"
=> 16 x (4 + 8 + 2 + 33 + 3)

==========> 800 bytes per 1.28 sec

. INS2 per 1.28 sec
16 X "$R1,<DIINS time><Length of data><data><CR><LF>"
=> 16 x (4 + 8 + 2 + 33 + 3)

==========> 800 bytes per 1.28 sec

. Loran per 1.0 sec
"$R3,<DIINS time><Length of data><data><CR><LF>"
=> 4 + 8 + 2 + 80 + 2

==========> 96 bytes per 1.0 sec

Sub-Total: Approx 41 MByte/Day

Filter Data breakdown is a follows:

. Filter (1 to 15) per 1.28 sec (highest rate)

15 x "$P<FilterID>1,<Display data><CR><LF>"
=> 15 x (4 + 8 + 2 + 11x8 + 2)
=> 1560 bytes

"$P<FilterID>2,<State data><CR><LF>"
=> 4 + 8 + 2 + (1+NumStates+NumStates+1)x8 + 2

FilterID - NumStates	=> size
1 - 14	=> 4 + 8 + 2 + (1+14+14+1)x8 + 2 = 256
2 - 23	=> 4 + 8 + 2 + (1+23+23+1)x8 + 2 = 400
3 - 23	=> 4 + 8 + 2 + (1+23+23+1)x8 + 2 = 400
4 - 19	=> 4 + 8 + 2 + (1+19+19+1)x8 + 2 = 336
5 - 19	=> 4 + 8 + 2 + (1+19+19+1)x8 + 2 = 336
6 - 21	=> 4 + 8 + 2 + (1+21+21+1)x8 + 2 = 368
7 - 21	=> 4 + 8 + 2 + (1+21+21+1)x8 + 2 = 368
8 - 21	=> 4 + 8 + 2 + (1+21+21+1)x8 + 2 = 368
9 - 21	=> 4 + 8 + 2 + (1+21+21+1)x8 + 2 = 368
10 - 17	=> 4 + 8 + 2 + (1+17+17+1)x8 + 2 = 304
11 - 17	=> 4 + 8 + 2 + (1+17+17+1)x8 + 2 = 304
12 - 17	=> 4 + 8 + 2 + (1+17+17+1)x8 + 2 = 304
13 - 17	=> 4 + 8 + 2 + (1+17+17+1)x8 + 2 = 304
14 - 18	=> 4 + 8 + 2 + (1+18+18+1)x8 + 2 = 320
15 - 18	=> 4 + 8 + 2 + (1+18+18+1)x8 + 2 = 320

=> 5056 bytes

"$P<FilterID>3,<Residual & Variance data><CR><LF>"
=> 4 + 8 + 2 + (1+NumMeas+NumMeas)x8 + 2

FilterID - NumStates => size
1 - 4 => 4 + 8 + 2 + (1+4+4)x8 + 2 = 88
2 - 10 => 4 + 8 + 2 + (1+10+10)x8 + 2 = 184
3 - 10 => 4 + 8 + 2 + (1+10+10)x8 + 2 = 184
4 - 6 => 4 + 8 + 2 + (1+6+6)x8 + 2 = 120
5 - 6 => 4 + 8 + 2 + (1+6+6)x8 + 2 = 120
6 - 8 => 4 + 8 + 2 + (1+8+8)x8 + 2 = 152
7 - 8 => 4 + 8 + 2 + (1+8+8)x8 + 2 = 152
8 - 8 => 4 + 8 + 2 + (1+8+8)x8 + 2 = 152
9 - 8 => 4 + 8 + 2 + (1+8+8)x8 + 2 = 152
10 - 4 => 4 + 8 + 2 + (1+4+4)x8 + 2 = 88
11 - 4 => 4 + 8 + 2 + (1+4+4)x8 + 2 = 88
12 - 4 => 4 + 8 + 2 + (1+4+4)x8 + 2 = 88
13 - 4 => 4 + 8 + 2 + (1+4+4)x8 + 2 = 88
14 - 6 => 4 + 8 + 2 + (1+6+6)x8 + 2 = 120
15 - 6 => 4 + 8 + 2 + (1+6+6)x8 + 2 = 120

=> 1896 bytes

==========> 8512 bytes per 1.28 sec

Sub-Total: ~~682.56~~ MByte/Day < 43

OVERALL TOTAL MAX DATA TO BE LOGGED IN ADM: ~~724~~ MByte/Day
690

APPENDIX E NEW SHINNADS TO DIINS INTERFACE REQUIREMENTS

E.1 Introduction

This appendix defines the new NMEA sentences required in order to produce the DIINS ADM from the XDM. These new sentences are required in order to implement the new HCI requirements for the ADM. The new sentences are used to communicate information between the HCI software and DIINS.

The sentence definitions presented in this appendix are preliminary, and will be subject to verification against the NMEA sentence formats defined in Reference[2-10]. In addition, the sentence formats may be modified during the detailed design in order to optimize the data transfer format. For example, rather than implement a form based sentence scheme as defined in this appendix, it may be more appropriate to partition the data in another manner. For example, all of the data associated with a particular sensor could be sent in a single sentence, e.g. its identifier, type, lever arm etc. The pros and cons of the alternative schemes will be examined as part of the software detailed design.

The following format definitions apply in the custom sentence definitions given in this section:

DRO Manufacturers mnemonic code for Defence Research Ottawa
*hh Sentence checkcode
<CR> Carriage Return
<LF> Linefeed

Note: Unit conversions required for display purposes (e.g. conversions to knots and yards) may be performed on the received sentences in the SHINNADS display software.

E.2 CMG, SMG, N Std Dev, E Std Dev and Est Error (Direction: DIINS->SHINNADS)

In the XDM, the North and East Covariances were sent to the SHINNADS display from the DIINS software. For the ADM, the DIINS software will take the square root of the North and East Covariances and send the standard deviations to SHINNADS, as indicated below.

$PDR01,x.x,T,x.x,M,x.x,N,x.x,E,x.x,C,c—c*hh<CR><LF>

where:

x.x,T = Course Made Good (CMG, degrees True)
x.x,M = Speed Made Good (SMG, meter/sec)
x.x,N = North Standard Deviation (meters)

x.x,E = East Standard Deviation (meters)
x.x,C = 95% Error Estimate (meters)
c---c = DIINS FDIR Status

Error Estimate Calculation:

The formula for computing the 95% Error Estimate from the north and east standard deviations is given in NATO Stanag 2178, Ed. 3, Para. 3.2.4.7. Based on this standard the following calculation will be performed:

Let SIGMAmax be the larger of the two standard deviations

SIGMAmax = max (SIGMAnorth, SIGMAeast)

and SIGMAmin be the smaller of the two:

SIGMAmin = min (SIGMAnorth, SIGMAeast)

Define

c = SIGMAmin / SIGMAmax

Then the 95% Error Estimate is:

R95 = (1.960790 + 0.004071*c + 0.114276*c^2 + 0.371625*c^3) * SIGMAmax

E.3 Sensor Available For Installation (Direction: DIINS->SHINNADS)

$PDRO3,tttt,ssss,*hh<CR><LF>

tttt = Sensor Type

Valid sensor types are as follows:

- INS1
- INS2
- GPS
- dGPS
- Loran

ssss = Sensor
Valid sensors are as follows:

- MK29 (INS)
- LTN90 (INS)
- MK49 (SINS) Note: Applicable to INS type.
- AshtechXII (GPS)
- TrimbleTANSPY (GPS)
- TrimbleVector (GPS)
- UofC (dGPS)
- LeicaMX94XX (dGPS)
- InternavLC360 (Loran)

E.4 Installed Sensor Model (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO4,tttt,ssss,*hh<CR><LF>

tttt and ssss = same as for PDRO3.

E.5 Selected Sensor (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO5,tttt,ssss,*hh<CR><LF>

tttt and ssss = same as for PDR03.

E.6 Sensor Lever Arm (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO6,tttt,ssss,xxxx.xx,yyyy.yy,zzzz.zz,*hh<CR><LF>

Where:

- tttt and ssss = same as for PDR03.
- x, y and z are the sensor lever arm coordinates in metres.

E.7 Sensor Communications Channel Parameters (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO7,tttt,ssss,ChID,BaudRate,NumBits,Parity,Stop,Handshake,*hh<CR><LF>

Where:

ChID: In range 00 - 99

BaudRate: {2400|9600|19200}

NumBits: {7|8}

Parity: {None|Odd|Even} i.e. Odd or Even

Stop: {1|2}

Handshake: {None|TBD}

E.8 FDI Parameters (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO8,MeasID,Sigma,ResidTest,FailFrac,FailTime,PassFrac,PassTime,*hh<CR><LF>

where:

- MeasID: alpha-numeric sensor measurement ID , e.g. X_GPS2_POS_MEAS
- MeasIndex dddd
- Sigma: f.f
- ResidTest: f.f
- FailFrac: f.f
- FailTime: f.f
- PassFrac: f.f
- PassTime: f.f

where "f.f" = floating point number

dddd = Integer index of parameter in the list of parameters

E.9 Data Logging

E.9.1 Selected Sensors For Logging (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

This message has a different format depending on its direction. All of the information is required to be sent from DIINS to the HCI, so for the DIINS -> SHINNADS direction the format is as follows:

$PDRO9,s1,L1,NBS1,s2,L2,NBS2,s3,L3,NBS3,s4,L4,NBS4,s5,L5,NBS5,*hh<CR><LF>

where:

s1 = INS1

s2 = INS2
s3 = GPS
s4 = dGPS
s5 = Loran

si = 0 : Not selected

si = 1: Selected

Li = Log interval (secs) for the ith sensor, format = f.f, a floating point number

NBSi = Number of bytes/second logged for the ith sensor

For the SHINNADS -> DIINS direction it is not necessary to send the log interval and number of bytes/sec, so the message in this case only contains the sensor types selected for logging, and the format is as follows:

$PDRO9,s1s2s3s4s5,*hh<CR><LF>

si = 0 : Not selected

si = 1: Selected

The order of the sensors is INS1, INS2, GPS, dGPS and Loran.

E.9.2 Selected Filters For Logging (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO10,s--s,b—b,*hh<CR><LF>

where:

s--s: indicates the selection status of each filter; s = 0 Not Selected, s = 1 Selected

Note: The order of the filters in s--s is Filter 1 to Filter n, e.g. n = 7 or n = 15.

b--b: indicates the number of bytes per second logged per filter, format = Integer

E.9.3 Logging Rate Scales (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO11,r0,r1,r2,r3,r4,r5,*hh<CR><LF>

where:

r0: Overall Rate Scale, format = f.f in range 0<= r<=1

r1 = INS1
r2 = INS2
r3 = GPS
r4 = dGPS
r5 = Loran

ri: Sensor Rate Scale, format = f.f in range 0<= r<=1

E.9.4 Filter Identifiers (Direction: DIINS -> SHINNADS)

$PDRO15,numOfFilters,FilterID1,FilterID2,...,*<CR><LF>

where :
numOfFilters = Indicates the number of filters to be logged.
FilterIDi = Text string representing the particular sensor combination used in the filter which is to be logged.

E.10 Failure Injection (Direction: SHINNADS->DIINS)

$PDRO12,Sensor,FailureMode,AbsRel,FailTime,Duration,*hh<CR><LF>

where:

Sensor: Sensor to have failure injected, from the list of Selected Sensors

FailureMode:	H = Hard S = Soft
AbsRel:	A = Absolute timing R = Relative timing
FailTime:	hhmmss.ss (absolute or relative)
Duration:	hhmmss.ss, failure injection duration

E.11 System Status

E.11.1 Processor and Filter Status (Direction: DIINS->SHINNADS)

$PDRO13,ProcLoad,SelFilter,*hh<CR><LF>

where:

ProcLoad:	f.f
SelFilter	integer number, in range 1 to number of filters.

E.11.2 Software Events (Direction: DIINS->SHINNADS)

$PDRO14,t,EventText,*hh<CR><LF>

where:

t:	hhmmss.ss, time of occurrence of the event
EventText:	character expression, with a maximum number of 40 characters, identifying the Software Exception, FDI Log or Filter Log events that occur.

E.12 High Rate Attitude Data Distribution (Direction: DIINS->Othe Ship Systems)

The following NMEA sentence format is assumed for use in the distribution of the high rate attitude information from DIINS to the other ship systems:

$PDRO16,INS,hhh.hh,srr.rr,spp.pp*hh<CR><LF>

where

INS = 1 for INS1, and 2 for INS2
hhh.hh = is the heading in degrees
s = sign, + or -
pp.pp = pitch
rr.rr = roll

E.13 INS CMG and SMG (Direction: DIINS->SHINNADS)

The course and speed information for the INSs will be contained in the following message format. There is no INS "Talker Mnemonic", so following the approach used in the XDM "HE" will be used as the mnemonic. To identify which INS the data comes from (i.e. INS1 or INS2), this message will always be preceded by the appropriate $HESTN sentence. The $HEVTG sentence will be broadcast at the same time as the $HEGLL and $HEHDT sentences for each of the INSs.

Note: The CMGs given here will be obtained from the resultant vector of the INS North and East velocities; whereas, the heading of the INSs in the $HEHDT sentence is the true heading of the INS (which may differ from its course). Both heading and CMG are available on the SINNADS display.

$HEVTG,x.x,T,x.x,M,x.x,N,x.x,K*hh<CR><LF>

where:

x.x,T = INS CMG (degrees True)
x.x,M =INS CMG (degrees Magnetic) (probably left undefined)
x.x,N =INS SMG (knots)
x.x,K =INS SMG (km/hr)

APPENDIX F DIINS ADM SENSOR CONFIGURATION DATA ORGANIZATION

F.1 Introduction

This appendix details the sensor configuration data scheme to be used for the DIINS ADM.

F.2 ADM Configuration Data Scheme

The solution adopted for the ADM to the problem of being able to easily add in a new sensor for the ADM, will be to place all of the information for the sensor (indeed for all of the sensors) into a single configuration data structure (SimSetup). Currently there may be areas of code which have to be changed when a new sensor is added to the configuration files. If there are such changes required in the XDM code, an attempt will be made to eliminate them during the ADM coding. The design goal has been to have all of the changes necessary to introduce a new sensor confined to a single data block in SimSetupData, as indicated in the following proposed data format:

```
--*****************************************************************************
----------[ Content-type: text/plain; name=SimSetupData ]----------

--This configuration data contains the definition of the sensors currently handled
--(Available Sensors) by the DIINS ADM. Separate data entry sections have been
--provided for both the primary and secondary sensors, as the default parameters
--contained in the data may differ between the two sensors of the same type. The default
--parameters will be used to initiate the data structures (e.g. list/array of data records) for
--the Available, Installed and Selected sensors, which will be used to interface with the
--user via the ADM HCI. Any modification via user inputs of the default values will be
--performed on these run-time data structures, will be re-displayed to the user and will be
--saved in non-volatile storage.

--*****************************************************************************
-- SPERRY MK29 INS1
--*****************************************************************************

Sensor Model:        SPERRY_MK29_INS1

DataDeltaT (sec):    1.28

Comms Port:          1
Baud Rate:           BAUD_9600
No. Bits:            BITS_8
Parity:              PARITY_NONE
```

Stop Bits: STOP_1
Handshake: HANDSHAKE_NONE

Lever Arm (X,Y,Z): 0.0 0.0 0.0 metres

Meas Model: LINEAR
Meas Correlation: UNCORRELATED
NoMeasTimeTrigger: 1.0 MINUTES

Data Log Rate (sec): 1.28

Installed: TRUE/FALSE, i.e. the sensor is physically installed in the operational system, and not just "available", that is available algorithms in the software.

SelectedForUse: FALSE (selected for use in the navigation solution)

LogRateScale: 1.0

LogInterval: 0.0

LogNomBytesPerSec: 0.0

SelectedForLogging: FALSE

```
--*****************************************************************************
-- Other Sensors TBD
--*****************************************************************************
```

TBD

```
--*****************************************************************************
```

F.3 Configuration Data Ada Package Structure

The following illustrates the kind of Ada package structure that might be appropriate to define the SensorConfigDataType, and to set and retrieve components of objects of that type to form the higher level data structures of the list/array of Available, Installed and Selected sensors planned to be used during the operation of the ADM.

```
--*****************************************************************************
--*****************************************************************************
-- PACKAGE SPEC SensorConfigData
-*****************************************************************************
```

```
--*************************************************************************

package SensorConfigData is

        type SensorConfigDataType is private

--*************************************************************************
-- CONSTRUCTORS
-*************************************************************************

        procedure Create (SensorCfgDataObject:...);

        procedure Set (TheSensorModel:...;InSensorCfgDataObject: ...);

        procedure Set (TheDataDeltaT:...;InSensorCfgDataObject: ...);

        procedure Set (TheCommsParams:...;InSensorCfgDataObject: ...);

        procedure Set (TheLeverArm:...;InSensorCfgDataObject: ...);

        procedure Set (TheMeasModel:...;InSensorCfgDataObject: ...);

        procedure Set (TheLogRate:...;InSensorCfgDataObject: ...);

--*************************************************************************
-- SELECTORS
-*************************************************************************

        function GetSensorModel (FromSensorCfgDataObject: ...) return ...;

        function GetDataDeltaT(FromSensorCfgDataObject: ...) return...;

        function GetCommsParams (FromSensorCfgDataObject: ...) return...;

        function GetLeverArm(FromSensorCfgDataObject: ...) return...;

        function GetMeasModel (FromSensorCfgDataObject: ...) return...;

        function GetLogRate (FromSensorCfgDataObject: ...) return...;

private

        type SensorConfigDataType is
                SensorModel: ...;
                DataDeltaT:...;
```

```
            CommsParams:...;
            LeverArm:...;
            MeasModel:...;
            LogRate:...;
        record

end record;

end SensorConfigData;

--*****************************************************************************
--*****************************************************************************
-- PACKAGE BODY SensorConfigData
--*****************************************************************************
--*****************************************************************************

package body SensorConfigData is
.
.
.
end SensorConfigData;

--*****************************************************************************
```

Note: The types not explicitly shown in the above package example, and indicated as "..." could be other user defined types, e.g.:

```
type CommsParamType  is
    record
    .
    .
    .
    end record;
```

APPENDIX G DIINS HCI CONTROL AND DISPLAY Ada CODE

G.1 Introduction

This appendix contains details of a Rational/Rose model generated to define the design requirements for the software required to interface to the software defined in Appendix C (i.e. the HCI, Display and Control software), but at the DIINS ADM Sensor Data Processor end of the RS422 communications link. The software on this interface will be used primarily by a maintenance engineer to gain access to the internal DIINS parameters during the setup or diagnostics of the DIINS ADM whilst on-board ship.

G.2 CSC Package

Figure G.2-1 indicates the main software package associated with the software defined in this appendix.

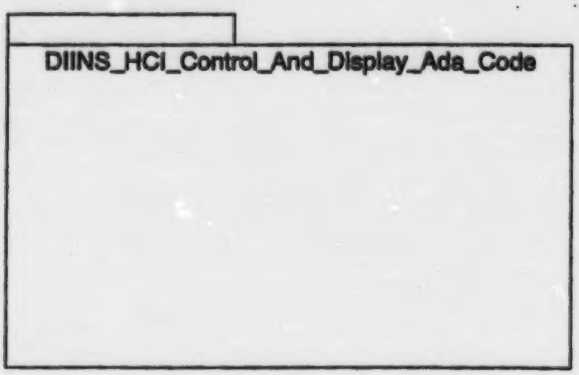


Figure G.2-1 DIINS HCI Control And Display Ada Code

Figure G.2-2 illustrates the lower level software packages associated with the Ada maintenance interface code.

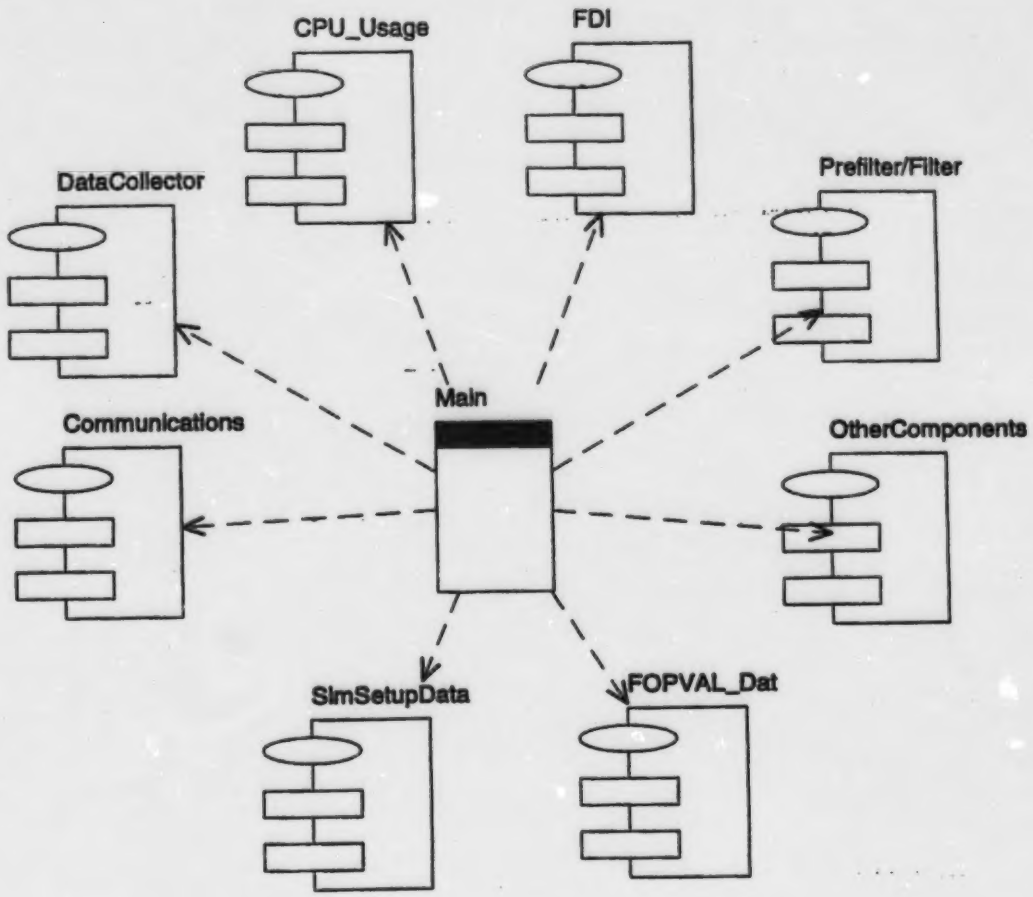


Figure G.2-2 Ada Interface Software Packages

G.3 DIINS HCI Control And Display Software CSC Classes

G.3.1 Data Collector Class

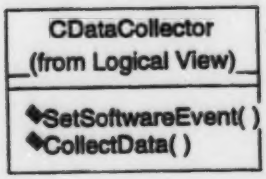


Figure G.3.1-1 Data Collector Class

G.3.2 CPU_Usage Class

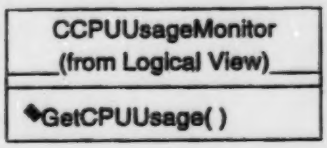


Figure G.3.2-1 CPU Usage Class

G.3.3 FDI Class

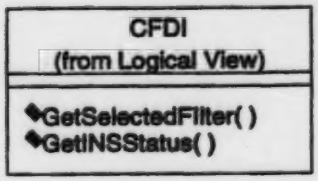


Figure G.3.3-1 FDI Class

G.3.4 Prefilter/Filter Classes

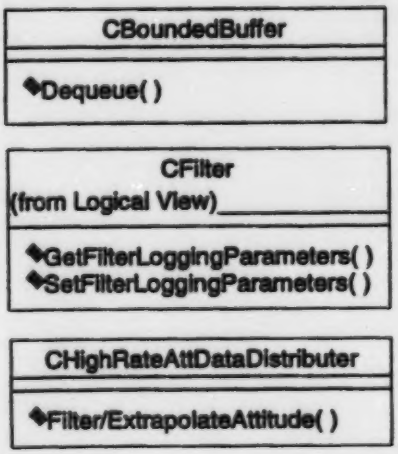


Figure G.3.4-1 Prefilter/Filter Class

G.3.5 Other Component Class

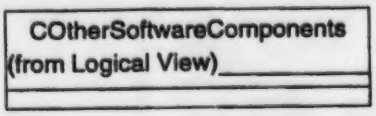


Figure G.3.5-1 Other Software Component Class

G.3.6 FOPVAL_Dat Class

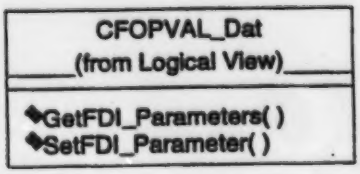


Figure G.3.6-1 FOPVAL Data Class

G.3.7 SimSetupData Class

Figure G.3.7-1 SimSetupData Class

G.3.8 Communications Classes

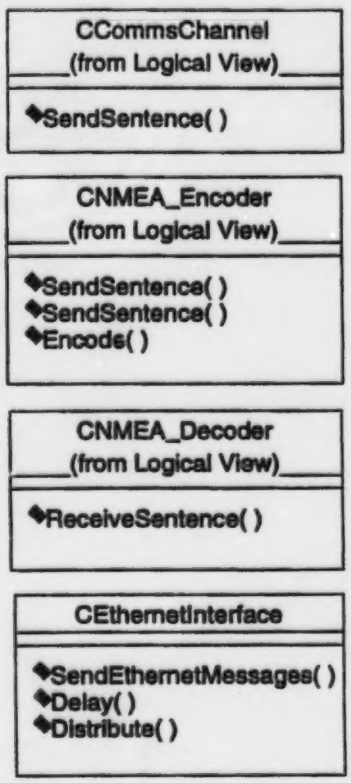


Figure G.3.8-1 Communications Classes

G.4 Ada Interface Code Behavior Sequence Diagrams

Note: The diagrams presented in this section correspond to the diagrams contained in Section C.4. The diagrams in this appendix supply the DIINS ADM information to the user interface as indicated on the diagrams in Section C.4, and receive information from the user interface as indicated in Section C.4.

G.4.1 Sensor Installation Sequence

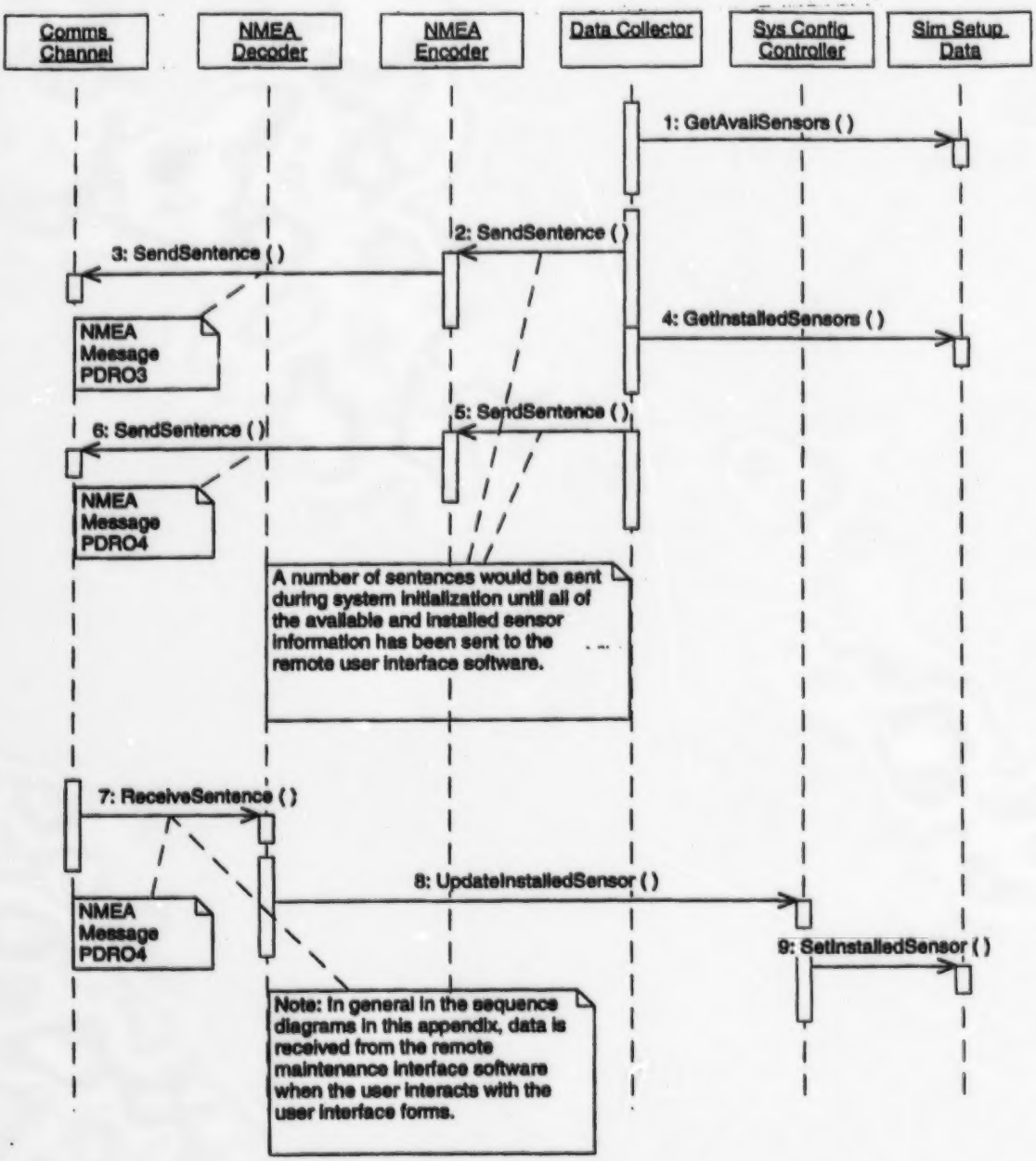

Comms Channel
NMEA Decoder
NMEA Encoder
Data Collector
Sys Config Controller
Sim Setup Data
1: GetAvailSensors ( )
2: SendSentence ( )
3: SendSentence ( )
NMEA Message PDRO3
4: GetInstalledSensors ( )
5: SendSentence ( )
6: SendSentence ( )
NMEA Message PDRO4
A number of sentences would be sent during system initialization until all of the available and installed sensor information has been sent to the remote user interface software.
7: ReceiveSentence ( )
8: UpdateInstalledSensor ( )
NMEA Message PDRO4
9: SetInstalledSensor ( )
Note: In general in the sequence diagrams in this appendix, data is received from the remote maintenance interface software when the user interacts with the user interface forms.

Figure G.4.1-1 Sensor Installation Sequence

G.4.2 Select Sensors Sequence

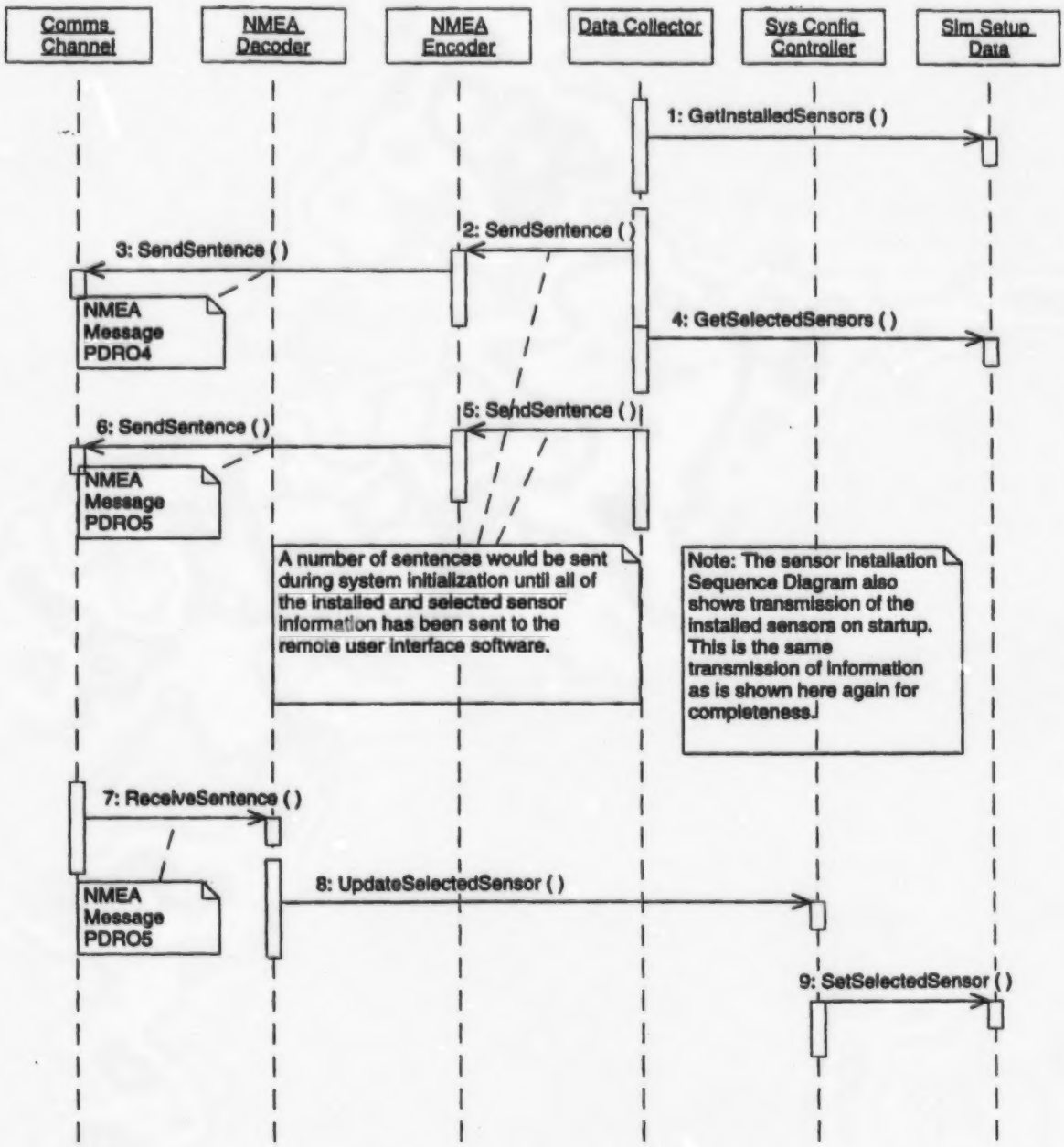

Comms Channel
NMEA Decoder
NMEA Encoder
Data Collector
Sys Config Controller
Sim Setup Data
1: GetInstalledSensors ( )
2: SendSentence ( )
3: SendSentence ( )
NMEA Message PDRO4
4: GetSelectedSensors ( )
5: SendSentence ( )
6: SendSentence ( )
NMEA Message PDRO5
A number of sentences would be sent during system initialization until all of the installed and selected sensor information has been sent to the remote user interface software.
Note: The sensor installation Sequence Diagram also shows transmission of the installed sensors on startup. This is the same transmission of information as is shown here again for completeness.
7: ReceiveSentence ( )
NMEA Message PDRO5
8: UpdateSelectedSensor ( )
9: SetSelectedSensor ( )

Figure G.4.2-1 Select Sensors Sequence Diagram

G.4.3 Lever Arm Setup Sequence

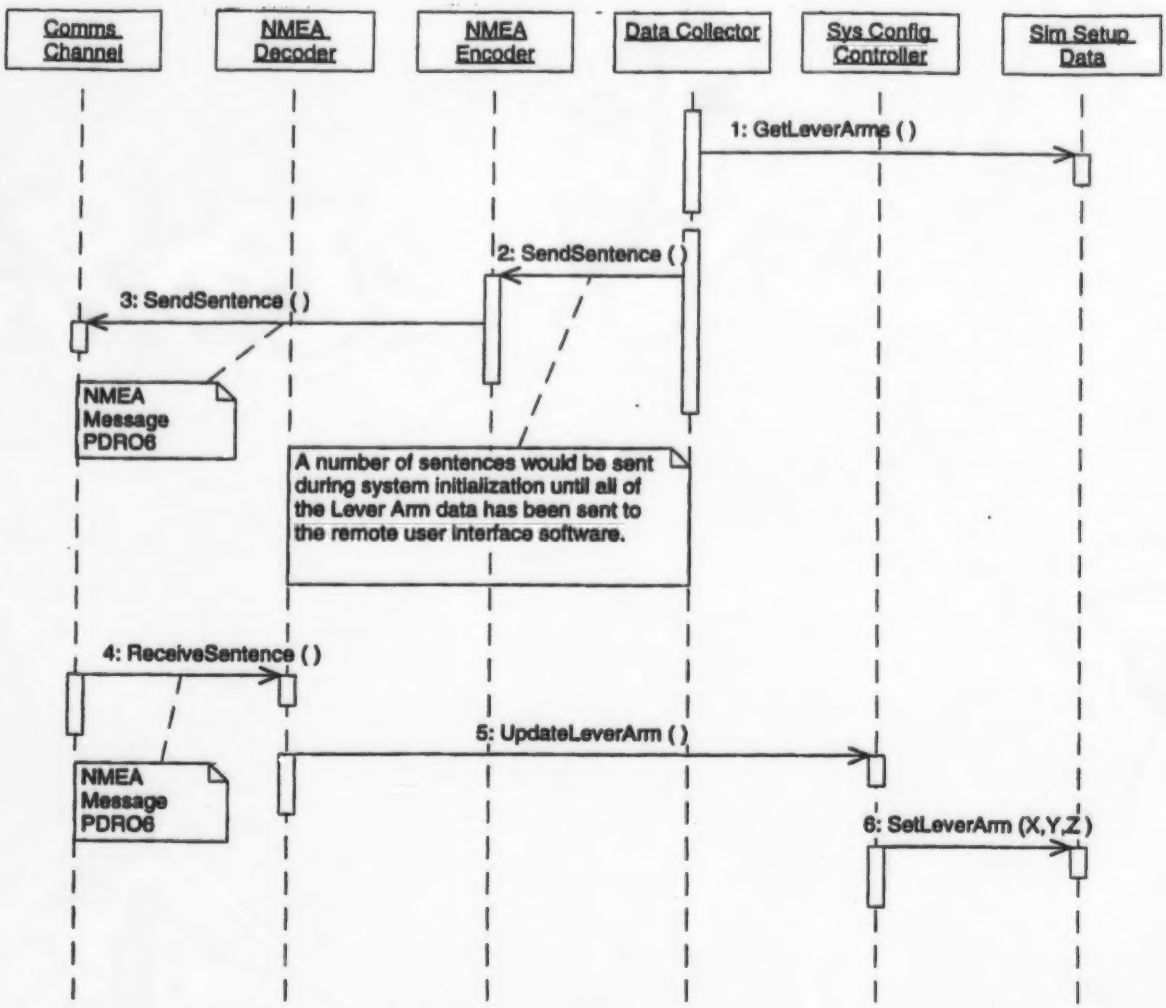

Comms Channel
NMEA Decoder
NMEA Encoder
Data Collector
Sys Config Controller
Sim Setup Data
1: GetLeverArms ( )
2: SendSentence ( )
3: SendSentence ( )
NMEA Message PDRO6
A number of sentences would be sent during system initialization until all of the Lever Arm data has been sent to the remote user interface software.
4: ReceiveSentence ( )
5: UpdateLeverArm ( )
NMEA Message PDRO6
6: SetLeverArm (X,Y,Z)

Figure G.4.3-1 Lever Arm Setup Sequence Diagram

G.4.4 Communication Parameters Setup Sequence

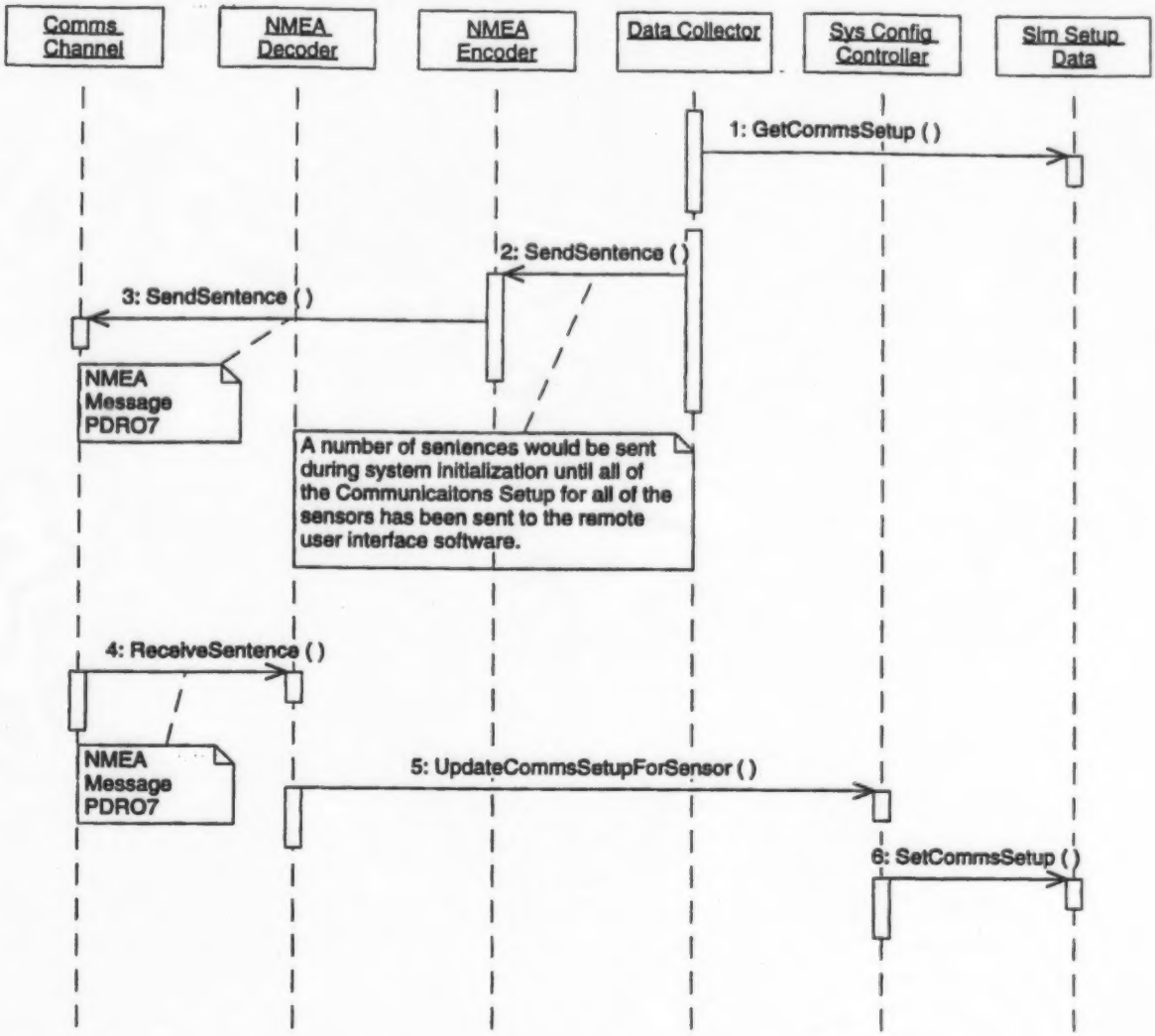

Comms Channel
NMEA Decoder
NMEA Encoder
Data Collector
Sys Config Controller
Sim Setup Data
1: GetCommsSetup ( )
2: SendSentence ( )
3: SendSentence ( )
NMEA Message PDRO7
A number of sentences would be sent during system initialization until all of the Communicaitons Setup for all of the sensors has been sent to the remote user interface software.
4: ReceiveSentence ( )
NMEA Message PDRO7
5: UpdateCommsSetupForSensor ( )
6: SetCommsSetup ( )

Figure G.4.4-1 Communications Parameters Setup Sequence Diagram

G.4.5 FDI Parameters Setup Sequence

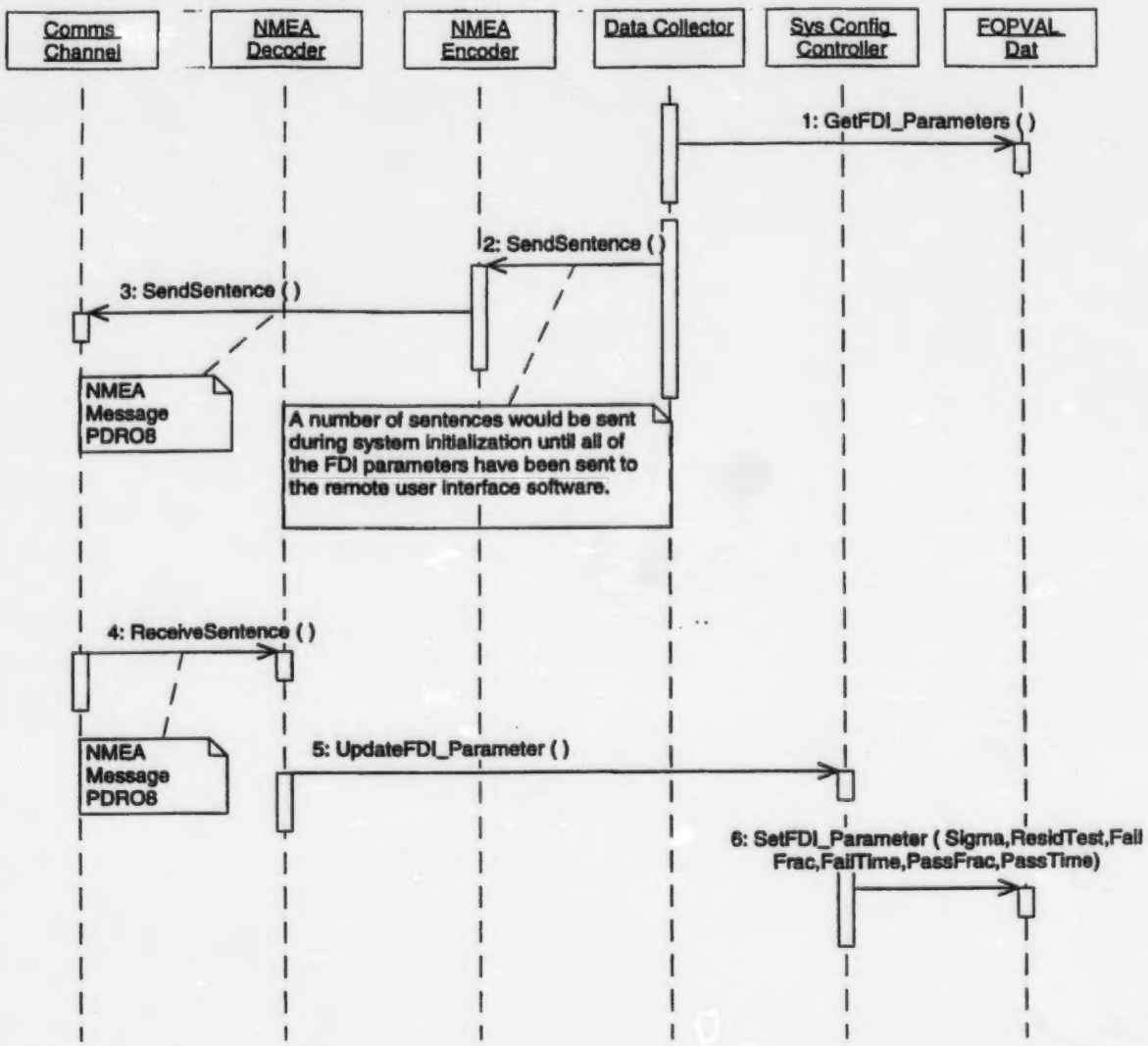

Comms Channel
NMEA Decoder
NMEA Encoder
Data Collector
Sys Config Controller
FOPVAL Dat
1: GetFDI_Parameters ( )
2: SendSentence ( )
3: SendSentence ( )
NMEA Message PDRO8
A number of sentences would be sent during system initialization until all of the FDI parameters have been sent to the remote user interface software.
4: ReceiveSentence ( )
NMEA Message PDRO8
5: UpdateFDI_Parameter ( )
6: SetFDI_Parameter (Sigma,ResidTest,Fail Frac,FailTime,PassFrac,PassTime)

Figure G.4.5-1 FDI Parameters Setup Sequence Diagram

G.4.6 Data Logging Setup Sequence

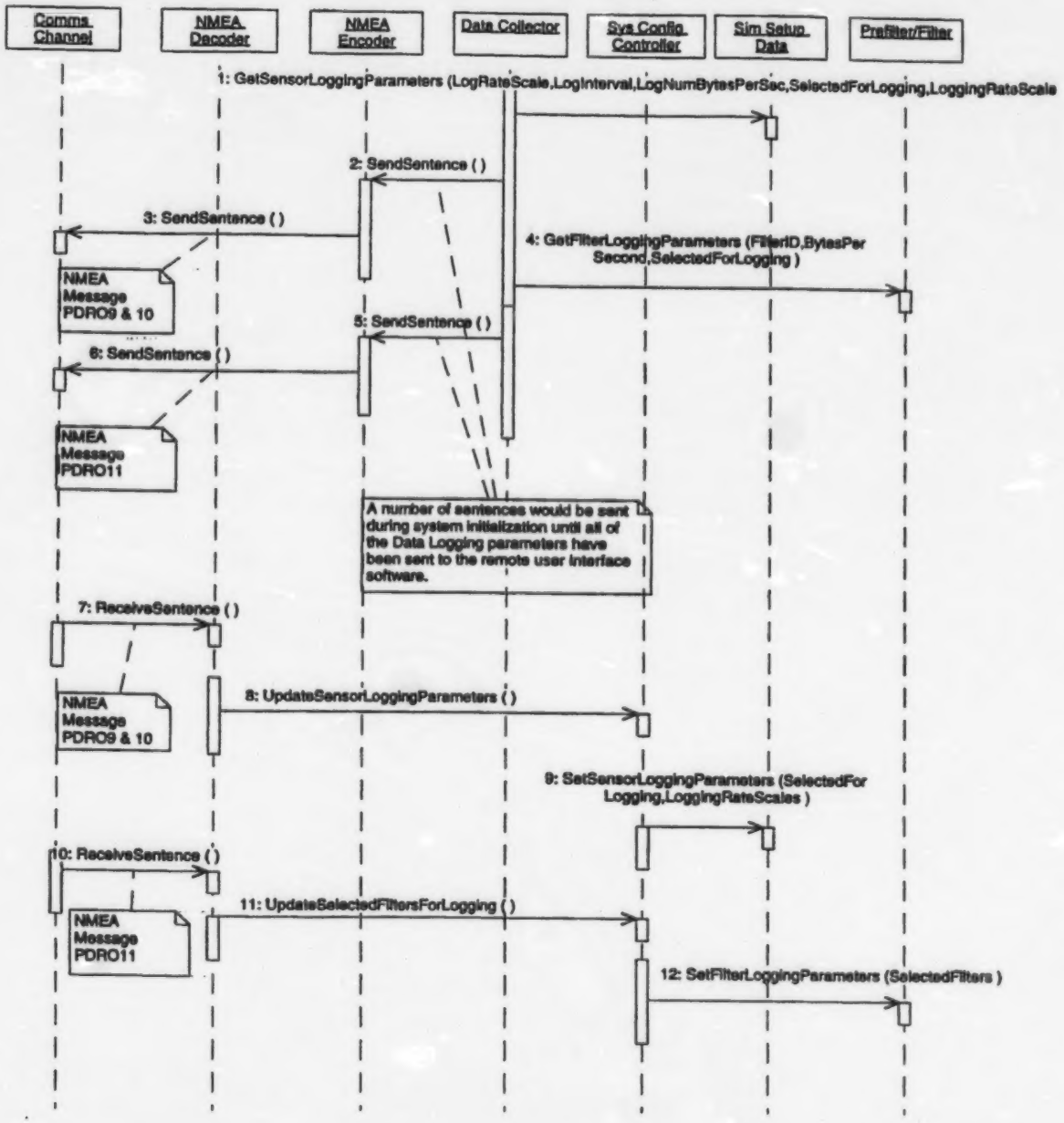

Comms Channel
NMEA Decoder
NMEA Encoder
Data Collector
Sys Config Controller
Sim Setup Data
Prefilter/Filter
1: GetSensorLoggingParameters (LogRateScale,LogInterval,LogNumBytesPerSec,SelectedForLogging,LoggingRateScale
2: SendSentence ( )
3: SendSentence ( )
4: GetFilterLoggingParameters (FilterID,BytesPer Second,SelectedForLogging)
NMEA Message PDRO9 & 10
5: SendSentence ( )
6: SendSentence ( )
NMEA Message PDRO11
A number of sentences would be sent during system initialization until all of the Data Logging parameters have been sent to the remote user interface software.
7: ReceiveSentence ( )
NMEA Message PDRO9 & 10
8: UpdateSensorLoggingParameters ( )
9: SetSensorLoggingParameters (SelectedFor Logging,LoggingRateScales)
10: ReceiveSentence ( )
11: UpdateSelectedFiltersForLogging ( )
NMEA Message PDRO11
12: SetFilterLoggingParameters (SelectedFilters)

Figure G.4.6-1 Data Logging Setup Sequence Diagram

G.4.7 Failure Injection Sequence

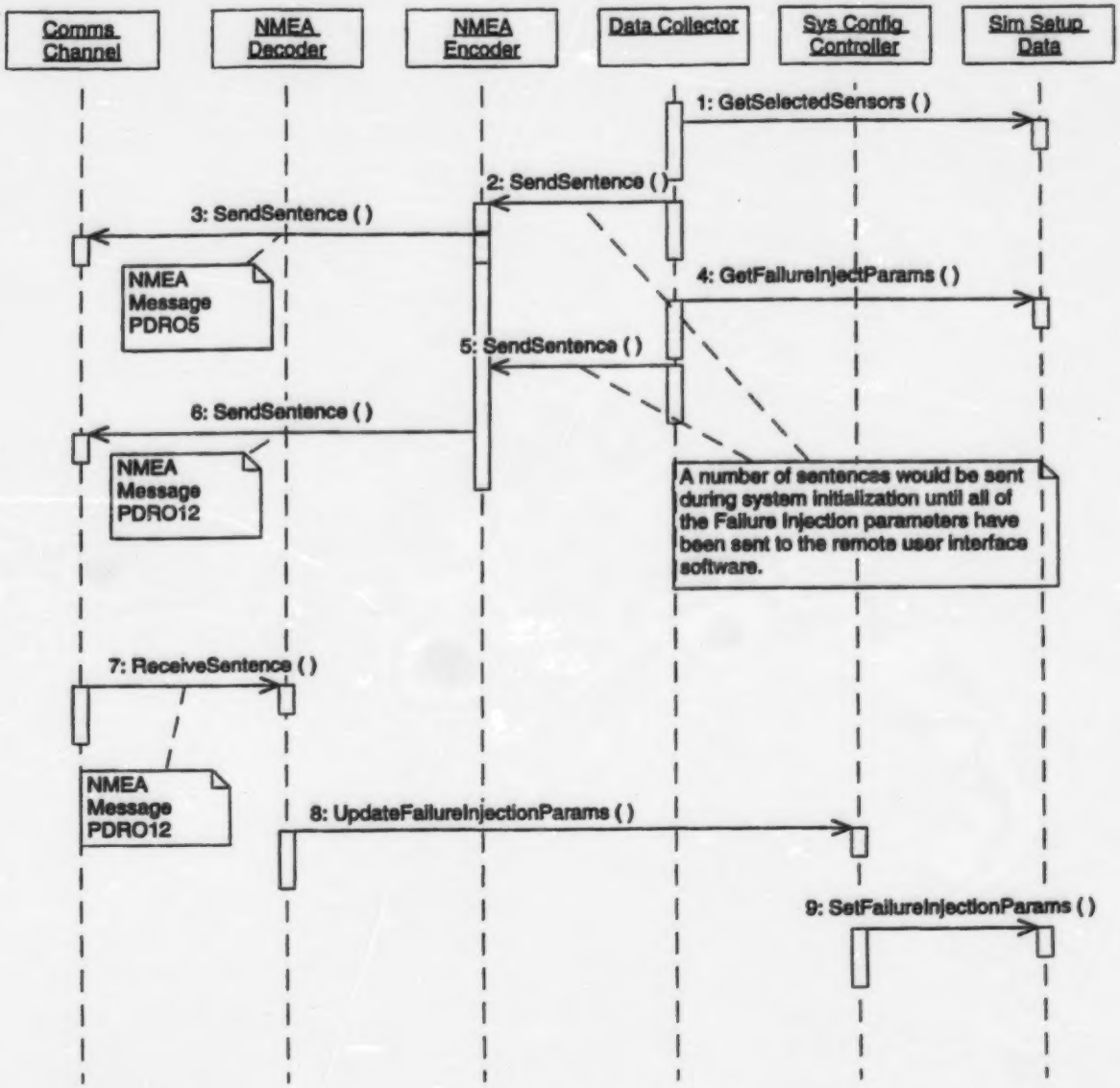

Comms Channel
NMEA Decoder
NMEA Encoder
Data Collector
Sys Config Controller
Sim Setup Data
1: GetSelectedSensors ( )
2: SendSentence ( )
3: SendSentence ( )
NMEA Message PDRO5
4: GetFailureInjectParams ( )
5: SendSentence ( )
6: SendSentence ( )
NMEA Message PDRO12
A number of sentences would be sent during system initialization until all of the Failure Injection parameters have been sent to the remote user interface software.
7: ReceiveSentence ( )
NMEA Message PDRO12
8: UpdateFailureInjectionParams ( )
9: SetFailureInjectionParams ( )

Figure G.4.7-1 Failure Injection Sequence Diagram

G.4.8 System Status Monitor Sequence

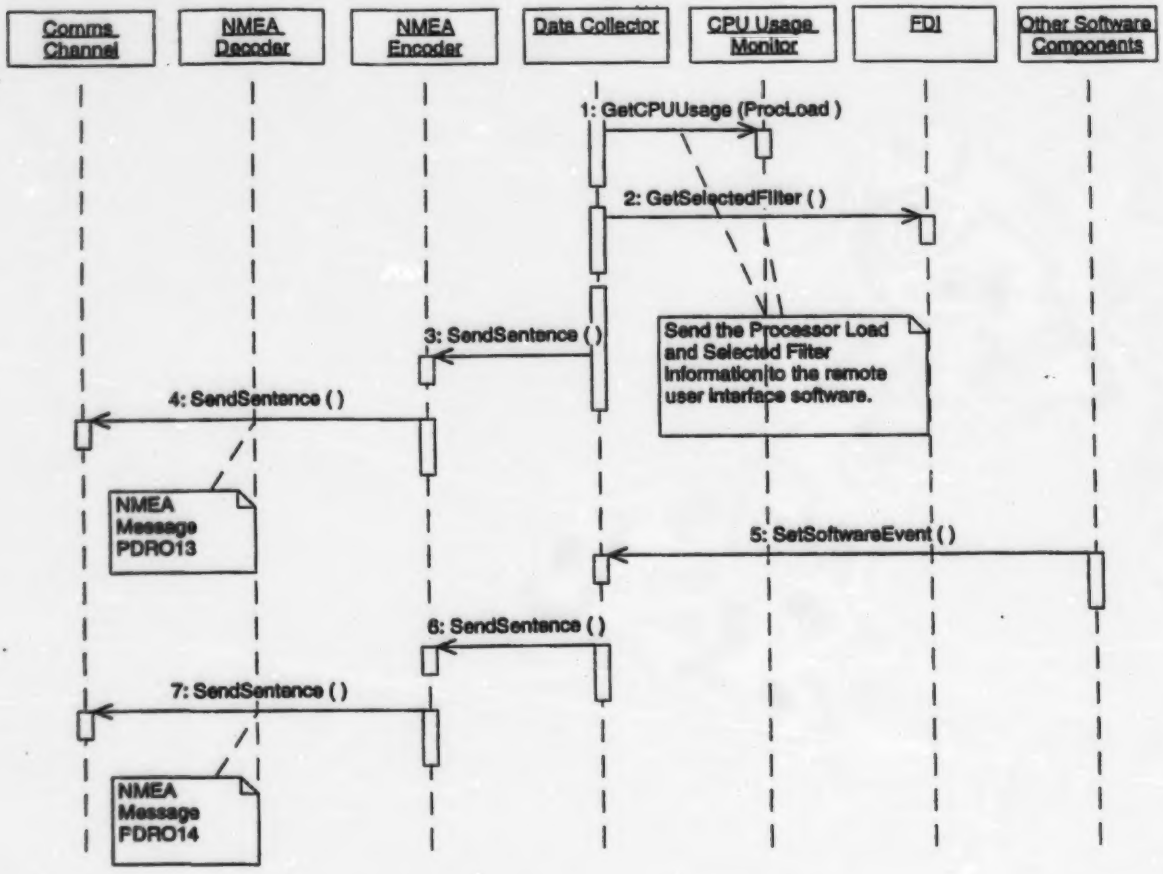


Figure G.4.8-1 System Status Monitor Sequence Diagram

APPENDIX H DIINS LIBRARY, SYSTEM AND UTILITY DIRECTORY CONTENT

The following lists the contents of the Library, System and Utility sub-directories of the DIINS software.

LIBRARY:

Assign.pkg
ChiSquaredFailure.pkg
DeFactor.ada.LC360
EnableFlagSetup.ada
F_ToPhi.ada
GeodeticLibrary.pkg
KalmanLibrary.ada.v03
KalmanLibrary.spc.v03
LeverArmCorrections.pkg
LinearSolverLibrary.ada
LinearSolverLibrary.spc
MakeDCM.pkg
MathLibraryExtension.ada
MathLibraryExtension.spc
MathUtilities.ada
MathUtilities.spc
NavCorrect.ada.v03
NavCorrect.spc.v03
PhiMxUD_MxMult.ada
PrefilterGeneric.spc
PropagatePhi.ada
PropagateTheP_Mx.ada.v03
PropagateTheStateVector.ada.v03
UD_Factor.ada.LC360
Update.ada

SYSTEM:

Ada_LAN_AddrB_167.ada
BBuffer.ada
BBuffer.spc
CPU_Usage.pkg
EnableFlagSetupLibrary.pkg
FDI_Global.ada
FDI_Pkg.spc

Install.spc
LoranStationData.pkg.LC360
MeasGlobal.spc.v03.LC360
NFS_167_RexB.ada
OutputResults.ada
PreInstanBBuffer.spc
RawSensors.spc.v03.LC360
ReadDataRates.ada
SimConfig.ada
SimConfig.spc
SimTypes.spc.v03.LC360
SimulationConfig.ada.LC360
SystemConfigGeneric.spc
SystemIndexGeneric.pkg.v03
TaskMaster.ada

UTILITY:

BinaryStandardIO.spc
BinaryStreamIO.pkg
C_Lang_Locals.pkg
Conversion.pkg
FileUtilities.ada
FileUtilities.spc
IRIG_B.ada
IRIG_B.spc
LumDebug.ada
LumDebug.spc
Scientific.ada
Scientific.spc
SparseInit.spc
StandardIO.ada
StandardIO.spc
Standard_E68.ada
Standard_E68.spc
String.ada
String.spc
Time.pkg

APPENDIX I 'WITHED' PACKAGES FOR DATA COLLECTION TASKS

The following indicates the list of packages withed by the various Ada units shown (using circled numbers) on the Data Collection Task 'Withing' diagram. The list numbers used in this appendix correspond to the numbers shown on the diagram.

All lists contain the packages System and Standard, which are not shown further below.

List 1:

- LumDebug
- RawSensors
- StringOperations
- DisplayDefs

List 2:

- LumDebug
- RawSensors
- StringOperations
- DisplayDefs
- MathLibraryExtension

List 3:

- RawSensors
- StringOperations
- DisplayDefs
- IRIG_B
- MathLibraryExtension
- Calendar

List 4:

- RawSensors
- StringOperations
- Unchecked_Conversion
- Float_Math_Library
- Text_IO
- Scientific_Constants
- IRIG_B
- SimConfig

List 5:

- RawSensors
- StringOperations
- Unchecked_Conversion

- Text_IO
- IRIG_B
- StandardIO

List 6:

- RawSensors
- StringOperations
- Unchecked_Conversion
- Scientific_Constants
- Text_IO
- StandardIO
- IRIG_B
- Long_Float_Math_Library

List 7:

- LumDebug
- RawSensors
- FDI_Pkg
- Text_IO
- StandardIO
- Unchecked_Conversion
- Scientific_Constants
- MathLibraryExtension
- Float_Math_Library
- AdaLAN_Socket
- SimConfig

List 8:

- LumDebug
- Conversion
- Interrupt
- InstallationParameters
- SimConfig
- Text_IO

List 9:

- LumDebug
- StringOperations
- Text_IO
- Conversion
- ATEX_Mailbox_Manager
- Interrupt
- IRIG_B

List 10:

- Conversion
- Text_IO
- Unchecked_Conversion

List 11:

- Conversion
- Text_IO
- Unchecked_Conversion
- ATEX_Mailbox_Manager
- IRIG_B

List 12:

- LumDebug
- StringOperations
- Unchecked_Conversion
- Text_IO
- Conversion
- ATEX_Mailbox_Manager
- Interrupt
- IRIG_B

List 13:

- LumDebug
- StringOperation
- Text_IO
- Conversion
- Unchecked_Conversion
- ATEX_Mailbox_Manager
- IRIG_B

UNCLASSIFIED
SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Prior Data Sciences Ltd.
240 Michael Cowpland Drive
Kanata ON K2M 1P6

2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

DIINS ADM Software Design Document Version 1.0 (U)

4. AUTHORS (Last name, first name, middle initial)

Jackson, G., Davis, T.

5. DATE OF PUBLICATION (month and year of publication of document)	6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)	6b. NO. OF REFS (total cited in document)
31 July 1998	138	11

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

CONTRACTOR REPORT

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

Defence Research Establishment Ottawa
Department of National Defence
Ottawa ON K1A 0Z4

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written)
Project 1bd17	W7714-6-0043

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)
DREO CR 1999-111	

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

UNLIMITED

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

This Software Design Document specifies the high level and detailed design of the software components and interfaces associated with the Computer Software Configuration Item (CSCI) for the Dual Inertial Integrated Navigation System (DIINS) Advanced Development Model (ADM) program.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

DIINS
Navigation Systems
Navigation Software
Software Design

#512195